USING SAS AT NIH - ON THE MAINFRAME BATCH MODE

Batch mode on the mainframe is a method of running SAS programs in which you prepare a file containing SAS statements and the necessary operating system control statements. While the program executes, control returns to the terminal or workstation environment where you can perform other tasks. Batch mode is sometimes referred to as "running in the background". The job output can be written to files or printed on an output device.

Office of Computing Resources and Services/SSS
Center for Information Technology
National Institutes of Health
Bethesda, Maryland 20892

Contents

Pages

CIT Services and Resources	1-2
Software Support on Various Computing Platforms	3-4
Description of SAS Products	5-6
A Glossary of Terms and Acronyms Used in this Handout	7
NIH Computer Center's System Manage Storage(SMS) Environment	8
Commonly Used WYLBUR commands to manage WYLBUR files	
NIH Computer Center is a Standard-Labeled Tape Facility	9
NIH Computer Center's WYLBUR Data Sets	
Edit Format	10
LRECL Format	11
SAS Reads Raw Data with LRECL Format on the Disk and the MSS	12
Estimate LRECL, BLOCKSIZE and SPACE Requirements	13
WYLBUR **ENTER DISKCALC** Command	
Estimate the Size of a SAS Data Set	14
Invoking a SAS Session on the NIH Mainframe	15
Batch Mode	
Interactive Mode	
A Sampling of JCL Statements Used in SAS Programs	
Input Samples	16-19
Output Samples	20-24
Access SPSS, BMDP, and DB2 Files	
Access SPSS from SAS	25
Access BMDP from SAS	26
Access DB2 Files	27-33
PROC BMDP Procedure	34-36
PROC CONVERT Procedure	37
Transporting SAS Files Between Operating Systems	38-39
A Sampling of Programs to Create and Import SAS Transport Files	40-42

Contents **Pages**

A Sampling of Programs to Transfer SAS Transport Files Between Operating Systems 43-46
- Download SAS Library 43
- Download SAS member-type CATALOG 44
- Upload SAS Library 45
- Upload SAS member-type CATALOG 46

How to… Transform SAS Data Set on MVS to SPSS file on PC 47
- Download an SPSS portable file to the PC 48

How to… Transform SPSS File on PC to a SAS Data Set on MVS 49-50
- Upload an SPSS portable file to MVS 49
- Convert standard SPSS file to a SAS data set on MVS 50

How to… Copy SAS Data Sets from one storage device to another 51

How to… Convert a version 5 SAS Library to version 6 52

How to… Convert a version 6 SAS Library to version 5 53

PROC SORT - Sorting Permanent SAS Data Sets 54

PROC CONTENTS and PROC DATASETS Procedures 55-57
- PROC CONTENTS procedure 55
- PROC DATASETS Procedure 56
  - CONTENTS, CHANGE, DELETE statements 57

SAS Format Library 58

A Sampling of Programs to Create, Use and Convert SAS Format Libraries 59

A Sampling of Programs to Create, Use and Convert Version 5 SAS Format Libraries 60

How to… Transform Version 5 SAS Format Libraries to Version 6 and Download to PC Environment 61

PROC Catalog Procedure 62
- How to… Add to or modify a SAS Format Library 62
- How to… Combine SAS Format Libraries 62
- How to… Use permanent SAS format library 63

PROC PRINTTO Procedure 64

Efficient SAS Programming - A Few Tips 65-66
- Use the LENGTH Statement to Reduce the Storage Space

How to Invoke the SAS Macro Facility 67

How to Use Curly Braces in a SAS Program at NIH 68

CIT Services and Resources

CIT'S central point of contact is the **Technical Assistance and Support Center (TASC)** of the Customer Services Branch (CSB), which provides technical and informational support to the NIH computing community and to CIT'S customers in other Federal agencies. The TASC help desk is staffed with full-time computer specialists. Users may also submit electronic mail and voice mail messages 24 hours a day and TASC will respond during regular operating hours.

Call:	4-DCRT (301-594-3278)
E-mail:	4DCRT@nih.gov
Fax:	301-402-7349
TDD:	301-496-8294
Visit:	Building 12A, Room 1011 Monday through Friday 7:30 A.M. to 5:00 P.M.
Mail:	12 South Drive MSC 5605 Bethesda, MD 20892-5605

World Wide Web access to CIT through the URL: http://www.cit.nih.gov/

TASC provides customer support for:

(1) MVS Systems

- Please refer to Interface 206, June 15, 1998, " The Evolution of WYLBUR" and "Planning is Underway for an MVS Standard System"
- MVS/ESA operating systems for both the MVS North and South systems
- **North System** (formerly ITS at the Parklawn Building)
  The North MVS System software includes JES2, TSO, ISPF, ACS, ACS WYLBUR, Model204, and relational databases.
- **South System** (formerly the MVS component of the NIH Computer Center)

This MVS component of the Enterprise Systems at the Computer Center is based on the IBM MVS/ESA (Multiple Virtual Storage/Enterprise System Architecture) Operating System with JES2 Multi-Access Spool.

The MVS System customer service areas include:

Processing Tasks

- access to batch processing
  as well as interactive systems(e.g., WYLBUR, ISPF, CICS, TSO)
- language compilers
- custom printing
- automatic data backup for desktop computers
- data security protection
- disaster recovery planning
- remote job entry

Connectivity Products for Access to the MVS System

- Terminal emulation and full connectivity software for PC and Macintosh clients for telnet and dialup connections.
- Supported software packages include MS-Kermit, PROCOMM PLUS, VersaTerm, PC/TCP and OnNet, and the Protocol Conversion Facility.

Networks

- NIHnet, a high-speed network backbone that interconnects NIH Ethernet LANs, the Computer Center mainframes (MVS and Helix Systems), and the Internet.
- LAN protocols supported for NIHnet connectivity are TCP/IP, AppleTalk, IPX, XNS, and DECnet.
- Dialup access to NIHnet is available through PARACHUTE
- Internet - and international collection of networks supported by major research institutions, which communicate with each other using TCP/IP protocols.
  The Internet offers file transfer, remote login(telnet) and mail services.

SILK Web Facilities

- The SILK (Secure Internet LinKed) Web technologies allow users to create and run their own Web servers using NIH Computer Center hardware and software.

Relational Database Systems

- DB2
- Oracle - Oracle RDBMS on the MVS and the Oracle server Digital AlphaServer 8400

IMS
- An information management system for interactive interrogation and maintenance of large centralized data bases(restricted access).

System Tape Library via MVS on the IBM

- The Tape Library, located in Building 12, Room 1100, maintains a large supply of magnetic tapes for use with MVS.
  The Tape Library may be contacted at (301) 496-6021.

Digital AlphaServer 8400

- Another component of the Enterprise Systems provides the base for a new environment for client/server(i.e. Unix-based) applications.

More detail information on the Enterprise Systems is available on the World Wide Web at http://cfb.dcrt.nih.gov/enterprise.html
To solve any problem you encounter when using the MVS System you may submit a PTR (Problem Tracking and Report System) through one of the following:

- WYLBUR **ENTER PTR** command
- World Wide Web access at http://cfb.dcrt.nih.gov/ptr.html
- E-mail to ptr@cu.nih.gov

(2) Advanced Laboratory Workstation (ALW) System

(3) Helix Systems

- Helix - a network communication system
- Convex - a computation server which runs scientific applications

In addition to the standard UNIX tools for software development, text formatting, and network communications, software packages include:

- Scientific Applications: GCG Sequence Analysis Package, Quest, BLAST, CHARMm, Gaussian, Mathematica, Prophet
- Biological Databases: GenBank, PIR, GCG, PDB
- Subroutine Libraries: IMSL - Mathematical and statistical routines
- Programming Languages: C and FORTRAN, Pascal, LISP, and C++

(4) Computer Training
The CIT Computer Training offers a wide variety of courses that enable users to make efficient and effective use of computing, networking, and information systems in their work. The training program is open to NIH employees and to all users of the CIT computing facilities. Course information and registration is available through CIT Home Page of the World Wide Web at **http://livewire.nih.gov/training/lookup.asp**

(5) Scientific Computing Resource Center (SCRC)
The Scientific Computing Resource Center is a shared-use computing facility for the hands-on evaluation and use of scientific software by NIH researchers.

(6) Customer Accounts
The Customer Account Office provides account and registration services to the NIH community and to customers in many other Federal agencies.

(7) Publications, Documentation and Software Distribution
CIT distributes publications and software to our customers. Documentation can be ordered by contacting TASC or online services through WYLBUR ENTER PUBWARE command or the Helix PUBWARE command.

(8) Statistical Support Staff provides technical support to NIH employees and all users of the Center for Information Technology(CIT) computing facilities. Web site: **http://statsoft.nih.gov/**

The Statistical Support Staff provides the following services: statistical advice, assistance on the interpretation of results, technical help on the use of the supported software, documentation for the supported software, training through the CIT Computer Training Program, access to the SAS Software for Windows, Mac and OS/2 through an NIH site license

You can contact the Statistical Support Staff at 301-594-3278(4-DCRT)
or send e-mail to 4dcrt@nih.gov.
Help is available Monday through Friday from 8:30 a.m. to 5:00 p.m.

Software Support on Various Computing Platforms

1) MVS/ESA Operating System

SAS/TUTOR Online Training Library, Base SAS, SAS/STAT, SAS/GRAPH, SAS/ETS, SAS/OR, SAS/FSP, SAS/AF, SAS/IML, SAS/ASSIST, SAS/CONNECT, SAS/INSIGHT, SAS/CALC, SAS/QC, SAS/TOOLKIT, SAS/ACCESS-DB2, SAS/ACCESS-ORACLE, SAS/SHARE

BMDP

A data analysis program library. Its programs perform descriptive statistics, tests for means, linear and nonlinear regression, log-linear models, maximum likelihood estimation, multivariate analysis, cluster analysis, survival analysis, and time series analysis.

SPSS

Provides capabilities for extensive file handling and data management tasks. Includes advanced statistical analysis procedures such as discriminant analysis, nonlinear and logistic regression analysis, and multivariate analysis of variance.

IMSL

This product is an extensive collection of FORTRAN-callable subroutines from Visual Numerics Inc. These routines perform mathematical and statistical computations.

GLIM

A system from the Numerical Algorithms Group Inc. for analysis of linear statistical models that can run in batch or interactive mode.

MSTAT1

This is a small collection of programs and subroutines for mathematical and statistical analysis. These programs were developed by CIT to run in batch mode.

LISREL

A program used to estimate the unknown coefficients of a set of linear structural equations.

2) **SAS System Software for Windows, Macintosh, and OS/2** is available to National Institutes of Health personnel through a site license between NIH/CIT and SAS Institute Inc.
NOTE:
Request for PC, Mac, or OS/2 SAS System Software must be made to the Technical Information Office. Please call 301-594-3278(4-DCRT).

SAS products available on Windows, Macintosh, or OS/2 through this site license are:

SAS/TUTOR Online Training Library (only CBT101), Base SAS, SAS/STAT, SAS/GRAPH, SAS/FSP, SAS/AF, SAS/IML, SAS/ASSIST, SAS/CONNECT, SAS/INSIGHT, SAS/CALC, SAS/EIS, SAS/LAB, SAS/ACCESS Interface to PC File Formats, SAS/ACCESS Interface to ODBC Software, SAS/QC, SAS/OR, SAS/ETS.

New to SAS System for the Windows, Mac, and OS/2 environments is the support for ODBC. ODBC is software architecture developed by Microsoft that provides a common Application Programming Interface (API). ODBC is a Microsoft standard to allow software packages that are compliant with the ODBC specifications to exchange information. Through the use of the ODBC engine, compliant software packages such as Lotus 1-2-3, Excel, Paradox, and others can now extract information directly from SAS data sets.

SAS/ACCESS Interface to PC File Formats provides an interface to PC software products. SAS System can be used to analyze and present data directly from popular PC file formats like dBase, DIF, LOTUS 1-2-3, and Microsoft Excel.

SAS/ACCESS Interface to ODBC provides all the power and flexibility of the SAS System that one can use to analyze and present data directly from mainframe platforms, UNIX and popular PC file formats.

3) **ALW System**
Base SAS, SAS/STAT, SAS/GRAPH, SAS/FSP, SAS/IML, SAS/CONNECT and SAS/INSIGHT

4) **Helix System(Convex server) and PC**
IMSL - Mathematical and statistical routines

5) **SUDAAN**
SUDAAN 7.50 for Windows can be installed as a ’stand-alone’ or as a ’SAS-callable’. SUDAAN is a single program consisting of a family of procedures used to analyze data from complex surveys and other observational and experimental studies involving cluster-correlated data. A complex sample may be multistage, stratified, or clustered. Many samples also have unequal probabilities of selection, or are drawn from finite populations. SUDAAN enables you to use survey data to obtain consistent estimates of population parameters and their standard errors in accordance with the sample designs. SUDAAN also produces consistent estimates of regression coefficients, descriptive statistics, and their associated standard errors for cluster-correlated and repeated measure data applications in clinical, epidemiological, toxicological, and behavioral research.

Description of SAS Products

To provide the most effective response to your SAS questions and problems, the Statistical Support Staff offers support for the following SAS products.
SAS/TUTOR Online Training Library:
Learn the SAS System as you use the SAS System with SAS Institute's library of online training courses. The SAS/TUTOR library on the mainframe currently includes six highly interactive courses. For more information, please see the CIT Computer Training brochure or call 301-594-3278.

BASE SAS:
This is the core of the SAS System. It is required for all other SAS System products. It contains the DATA step and the fundamental procedures that are needed for working with SAS data sets.

SAS/STAT:
Provides procedures for regression analysis, analysis of variance, categorical data analysis, multivariate analysis, discriminant analysis, scoring procedures, and survival analysis.

SAS/GRAPH:
Produces high-resolution graphics including several kinds of plots and charts, geographic maps, and a number of three-dimensional graphs.

SAS/ETS:
Provides procedures for time series analysis, financial report writing, linear and nonlinear systems modeling, and forecasting.

SAS/OR:
Provides a set of procedures for operations research and project management. Included in its capabilities are linear programming, Gantt charts, activity networks, and decision analysis.

SAS/FSP:
A component of the SAS System that provides interactive facilities for data entry, editing, and retrieval, printing form letters and reports, browsing external files, and building data entry and editing applications. SAS/FSP
includes several procedures and a programming language (SAS Screen Control Language) for building applications. Screen Control Language (SCL) is used to write instructions that control how an application responds to user actions.

SAS/AF:
This is an interactive applications development facility that enables programmers to create interactive windowing applications. It consists of a developmental environment that includes, PROC BUILD, a programming language (SAS Screen Control Language), and an interactive full-screen debugger. Screen Control Language (SCL) is used to write instructions that control how an application responds to user actions. Object-Oriented FRAME entries, a new component of SAS/AF software enables the developers to build SAS/AF applications as graphical user interfaces. A set of predefined graphical and text objects can be used to design or paint each application window.

SAS/ASSIST:A windowing facility that lets the user run the SAS System by making selections from a series of menus. Enables users to utilize SAS with very little SAS training.

SAS/CONNECT:
Enables communications between SAS Systems running on multiple remote hosts.

SAS/INSIGHT:
Explore data analysis through a variety of interactive graphic displays.

SAS/CALC:
A interactive windowing environment that enables you to create electronic spreadsheets. You use SAS/CALC software to enter data into a spreadsheet, define formulas to compute values, generate reports and graphics, create SAS data sets, and fetch data from SAS data sets.

SAS/QC:
Provides specialized tools for quality control and design of experiments.

SAS/TOOLKIT:
Used to write SAS procedures, functions, CALL routines, formats, and informats.

SAS/SHARE:
This product allows concurrent access to SAS data sets for reading and updating. With SAS/SHARE, multiple users can gain simultaneous update access to SAS files.

SAS/IML:
An interactive matrix facility with an extensive set of mathematical and matrix operators.

SAS/ACCESS - DB2:
The SAS/ACCESS interface to DB2 can be used to analyze and present data directly from DB2.

SAS/ACCESS - Oracle:
The SAS/ACCESS interface to ORACLE can be used to analyze and present data directly from ORACLE.

A Glossary of Terms and Acronyms Used in this Handout

Job Control Language (JCL) is a stream of statements used in batch mode to identify your job to the operating system and inform the operating system of the resources such as data sets, time, and memory that your job needs. JCL must be typed in upper case, begin in column 1, and extend no farther than column 71.
A job submitted to the operating system for batch processing begins with a **JCL JOB** statement and ends with a **JCL SYSIN** (//SYSIN DD *) statement

JCL JOB statement: //iii JOB (aaaa,box,class,cpu,lines),lastname
where the words shown in lower case are replaced with the user's information:

iii	registered initials	**box**	box number(optional)
aaaa	CIT user account	**class**	job class
lastname	user's last name	**cpu**	cpu time(optional)
		lines	maximum lines(optional)

The job class determines the resources needed for the job. Small jobs not requiring tapes (but possibly using the Managed Storage System (MSS)) get fast turnaround time from class A or class E. Class B or C is needed to mount a tape. Other parameters, such as maximum CPU time or maximum lines, may appear after the job class. For more details, see the *Computer Center User's Guide*, and *Batch Processing and Utilities at NIH* available from the Technical Information Office.

DD statement	a data definition statement that describes an operating system data set to the operating system, including the resources needed for the data set. The way the program can use a data set depends on the parameters in the DD statement.
ddname	a name(single word of user's choice) defined by a JCL DD statement. The ddname is how the operating system references the file. The ddname can contain one to eight characters; the first character must be a letter or a national character.
libref	the name used to identify a SAS data library to the SAS System. You assign a *libref* with a LIBNAME statement or with the ddname in the DD statement. The *libref* is the first-level name of a two-level name. For example, PERM is the *libref* in the two-level name PERM.CLASS.
fileref	the name used to identify an external file to the SAS System. You assign a *filref* with a FILENAME statement or with the ddname in the DD statement. The ddname and the *fileref* are the same. The *fileref* is how the SAS System references the file; the ddname is how the operating system references the file.

The fileref or the libref must be a valid SAS name to 'link to' or to 'point to' the file.

dsname	operating system data set name
nnnnnn	serial number of a tape
seqnum	sequence number of a data set on tape(default is 1)
lab	type of label on tape (SL, NL, AL--default is SL)
len	maximum line length (used in the LRECL=*parameter*)
b	block size (used in the BLKSIZE= *parameter*)
s1	primary space allocation (used in the SPACE= *parameter*)
s2	secondary space allocation (used in the SPACE= *parameter*)

NIH Computer Center's System Managed Storage(SMS) Environment

The SMS environment operates within an all-cataloged public disk storage environment. The direct access storage is on RAMAC (3390)online disks. The various types of storage have different restrictions on the length of storage time and backup service. The FILE and TMP management classes provide permanent and short-term storage for most data sets. The MSS management class provides slightly less expensive storage for data sets that do not require backup.

Commonly used WYLBUR commands to manage WYLBUR files:

SHOW DSNAMES ON CATALOG - searches the user's catalog and lists the user's cataloged data sets and the management class on which it resides.

SHOW DSNAMES ON CATALOG FULL - searches the user's catalog and lists the user's cataloged data sets, the management class and the actual *volser*, BLKSIZE, LRECL, DSORG, RECFM, SPACE and RACF protection information for the data sets.

SHOW DSNAMES ON FILE - searches the user's catalog and lists the user's cataloged data sets on the FILE management class.

SHOW DSNAMES ON MSS - searches the user's catalog and lists the user's cataloged data sets on the MSS management class.

SHOW DSNAME dsname - determine if a data set on the disk is stored in EDIT format

Changing Management Classes with SMS:
Another practical feature introduced with SMS managed data sets is the ability to switch a data set from one management class to another without having to physically move or copy the data set.

General Form of the TSO command:

```
      ALTER ’aaaaiii.dsname’ MANAGEMENTCLASS(class-name)
```

Specify TMP or MSS or FILE for *class-name.*

For example, enter the following TSO command:

```
ALTER ’aaaaiii.ONE’ MANAGEMENTCLASS(MSS)
```

… puts data set ONE into the MSS management class

or

```
ALTER 'aaaaiii.TWO' MANAGEMENTCLASS(FILE)
```

… puts the data set TWO into the FILE management class.

Consequently, just the class is changed.

NIH Computer Center is a Standard-Labeled Tape Facility

The standard-labeled tape 3480 cartridge tape is the NIH Computer Center’s standard tape. This tape has 18 tracks and stores information at 38,000 BPI. If you are interested in using another type of tape, refer to the *Computer Center’s User GUIDE*.

Because tape drives are not available on all of the Computer Center’s subsystems, a **/*ROUTE XEQ** statement must be included in any job that uses a tape.

General Form: **/* ROUTE XEQ** *resource-name*

where "resource-name" is a name specifying the type of tape drive required

Whenever a job uses a preassigned NIH tape or a Special tape, the volume serial number must be included on a /*MESSAGE statement

General Forms: **/* MESSAGE** *nnnnnn,R*
/* MESSAGE *nnnnnn,w*
/* MESSAGE *nnnnnn,S*

"nnnnnn" is the volume serial number of the tape requested
"R" reading only
"W" writing only
"S" is included only for (non-NIH) tapes.

Please refer to the document titled ***USING TAPES AT NIH.*** This manual is written to provide guidelines for using tapes on the IBM System at NIH

To have a tape assigned, call the Information Media Library at (301)496-6021.

NIH Computer Center’s WYLBUR Data Sets

WYLBUR data sets are stored in one of two formats on the disk and the MSS.

EDIT format or LRECL format:

EDIT FORMAT
The EDIT format is the standard format used by WYLBUR to store data sets.
EDIT format WYLBUR data sets are compressed to save space, while LRECL format data sets are not. EDIT format is the default storage format. If a WYLBUR data set is saved using the WYLBUR **SAVE** command, it is stored in EDIT format.

```
   ? SAVE AS dsname
```

EXAMPLE:

```
   ? COLLECT CLEAR
       1.   ? 4 5
       2.   ? 0 1
       3.   ? 2 6
   ? ***
   ? SAVE AS RAWDATA1
   ’RAWDATA1’ SAVED AND CATALOGED ON FILE
```

To determine if a WYLBUR data set on the disk is stored in EDIT format, use WYLBUR’s **SHOW DSNAME** command.

```
   ? SHOW DSNAME dsname
```

EXAMPLE:

```
   ? SHOW DSNAME RAWDATA1
   FILE

      RAWDATA1 ON DSA106

      UNIT=FILE, DEVTYPE=3010200F, NO. OF VOLUMES=1
      CREATED 06/03/97, LAST USED 06/03/97
      1 TRACK (1 USED), NO.  OF AREAS=1, SECONDARY SPACE=1 TRACK
      DSORG=PS, RECFM=U, LRECL=11476, BLKSIZE=11476
      KEYLEN=0, RKP=0, OPTCD=C(20)
      EXPIRATION DATE=00/00/00, NO PASSWORD
```

If the WYLBUR data set is stored in EDIT format, this command will report **RECFM=U, LRECL=11476, and BLKSIZE=11476.**

To determine if a WYLBUR data set on the MSS is stored in EDIT format, use WYLBUR’s

```
   ? SHOW DSNAME dsname on MSS
```

If the WYLBUR data set is stored in EDIT format, this command will report **RECFM=U, LRECL=11476, and BLKSIZE=11476.**

LRECL Format

To save a WYLBUR data set on the disk in LRECL format, add the LRECL option to WYLBUR’s **SAVE** command.

```
   ? SAVE AS dsname LRECL=len
```

where ’len’ is greater than or equal to the length of the longest data line.

EXAMPLE:

```
   ? COLLECT CLEAR
       1.   ? 4 5
       2.   ? 0 1
       3.   ? 2 6
   ? ***
   ? SAVE AS RAWDATA1 LRECL=6
   ’RAWDATA1’ SAVED AND CATALOGED ON FILE
```

To determine if a WYLBUR data set on disk is stored in LRECL format, use WYLBUR’s **SHOW DSNAME** command.

EXAMPLE:

```
   ? SHOW DSNAME RAWDATA1
   FILE

      RAWDATA1 ON DSA106

      UNIT=FILE, DEVTYPE=3010200F, NO. OF VOLUMES=1
      CREATED 06/03/97, LAST USED 06/03/97
      1 TRACK (1 USED), NO.  OF AREAS=1, SECONDARY SPACE=1 TRACK
      DSORG=PS, RECFM=FB, LRECL=6, BLKSIZE=11472
      KEYLEN=0, RKP=0, OPTCD=C(20)
      EXPIRATION DATE=00/00/00, NO PASSWORD
```

SAS Reads Raw Data with LRECL Format on the Disk and the MSS

SAS **does not** read raw data files saved in EDIT format. Therefore, if your data set is saved in EDIT format, you will need to apply one of the following:**1**) Use WYLBUR's **USE FROM** and **RESAVE** commands on the data set

```
? USE FROM dsname
? RESAVE LRECL=len
```

where 'len' is greater than or equal to the length of the longest data line.

OR

2) Include the **EDSIN** utility in the JCL

a) Reading raw data on the disk or the MSS in EDIT format with data lines **not extending** past column 80

```
//  EXEC EDSIN,NAME='aaaaiii.dsname'
//  EXEC SAS
//ddname DD DSN=&INPUT,DISP=(OLD,DELETE)
```

b) Reading raw data on the disk or the MSS in EDIT format with data lines **extending** past column 80

```
//  EXEC EDSIN,NAME='aaaaiii.dsname',LRECL=len,BLKSIZE=b
//  EXEC SAS
//ddname DD DSN=&INPUT,DISP=(OLD,DELETE)
```

In this instance the block size (BLKSIZE) and logical record length (LRECL) must be determined.

Estimate LRECL, BLOCK SIZE and SPACE Requirements

To output raw data from SAS, the logical record length, block size, and space allocations must be determined.

- secondary space allocation ('s2') can be 1 or approximately 10% of the primary space allocation('s1').
- LRECL equal to an integer greater than or equal to the length of the longest record.

Example: Compute BLKSIZE… given LRECL=133 and output 500 records to a raw data file.

WYLBUR's ENTER DISKCALC Command

```
? Enter Diskcalc

DISKCALC Command Procedure:

 Select a function from the following menu (or strike ENTER):
 (For fixed length, NON-keyed data sets)
 (Function 2 is applicable to all data set types)
  1   What is the Computer Center recommendation for BLKSIZE for a given LRECL?

  2   Compute how many physical records (blocks) may be stored on a track or cylinder with a given
      BLKSIZE.

  3   Show records and blocks per track and cylinder for a selection of possible BLKSIZES for a given
      LRECL.

  4   Compute total space needed to contain a specified number of records of a given LRECL for a
      selection of possible BLKSIZES.

  5   Change default disk model (currently 3390-3, 3,339 cylinders per volume, 15 tracks/cylinder,
      56,664 bytes per track).
      END    Return to WYLBUR.

      Select a function (or press ENTER to see the function menu): 4Enter the LRECL (logical record
      length) of the record (or press Enter):
      LRECL: 133

      Enter the number of logical records (or press ENTER):
      Number of records: 500
      ( press ENTER )

      Space required for 500 records with LRECL=133 on a 3390-3 volume which contains approximately
      3,339 cylinders per volume.
                     Number of      Number of
       BLKSIZE       Tracks         Cylinders
      -----------------------------------
      3,724           2
      4,123           2
      4,522           2
      5,054           2
      5,719           2
      6,517           2
      7,448           2
      8,778           2
      10,773          2
      13,566          2  <=NIH Recommendation
      18,354          2
      27,930          2
   Select a function (or press ENTER to see the function menu): END
   DISKCALC  terminating.
```

Result: NIH's recommendation is: **LRECL=133, BLKSIZE=13,566.**

Estimate the Size of a SAS Data Set

When storing a SAS data set, the record format (FB), the logical record length (LRECL), and block size (BLKSIZE) are set automatically.
However, you **must determine** the space allocation.
General Form of the SPACE parameter: SPACE=(TRK,(s1,s2),RLSE)

Space can be requested in terms of tracks or cylinders.
TRK specifies that the space is to be allocated in units of tracks.
CYL specifies that the space is to be allocated in units of cylinders.
(15 tracks per cylinder.)
RLSE indicates that space not used by the WYLBUR data set is to be released; RLSE should be specified whenever possible.

You can follow these steps to obtain a rough estimate of how much space you need on the disk or the MSS to store SAS data sets.

Step 1. To determine the length of each variable in the SAS data set, create a SAS data set that contains 1 observation.

```
FILENAME  IN1  'aaaaiii.dsname'  ;
DATA TEST;
  INFILE IN1  obs=1 ;
  INPUT  variable list   ;
PROC CONTENTS;
```

Step 2. Based on the results of the PROC CONTENTS procedure, determine the total length of all variables for each obseration.
(*Length of each variable can range from 2 to 8 for numeric variables and from 1 to 200 for character variables.)*

Step 3. Apply the following formula to estimate the primary space parameter.

Let X = (total length of all variables)*(total number of observations)
(The "total length of all variables" is specified as the "observation length" in the PROC CONTENTS Output.)

Let Y1 = X/6144. Let Y2 be Y1 rounded up to the next integer.
Let Z = Y2/7. The primary space allocation 's1' is Z rounded up to the next integer.

Example:
Suppose you have 12 numeric variables (each stored in 8 bytes) and 5000 observations.

```
X  = 8*12*5000 = 480,000
Y1 = 480,000/6144 = 78.125
Y2 = 79
Z  = 79/7 = 11.29
s1 = 12 tracks
```

Result: SPACE=(TRK,(12,1),RLSE).

A 'rule of thumb'
The primary space parameter('s1') - set as the larger of the two parameters.
The secondary space parameter ('s2') - set approximately 10% of the primary space allocation(10% of the 's1' parameter).

Invoking SAS on the Mainframe at NIH

1. Batch Mode
When a SAS program is submitted to the MVS operating system for batch processing the SAS program begins with the Job Control Language (JCL).

SAS Version 6 is the default MVS version of SAS at NIH. Therefore, the following EXEC statement is used to invoke SAS

```
      //  EXEC  SAS
```

The following JCLLIB statement is required to 'link' the SAS procedures located in our private library of cataloged procedures.

```
      //  JCLLIB ORDER=ZABCRUN.PROCLIB
```

Example:
The JCL statements used to submit a SAS program for batch processing when the input data immediately follows a CARDS statement in the SAS program.

```
      //  (JOB statement)
      //  JCLLIB ORDER=ZABCRUN.PROCLIB
      //  EXEC SAS
      //SYSIN DD *

      DATA TEST;
        INPUT variable list ;
      CARDS;
      (data lines)
      ;
```

When submitting a SAS program for batch processing, be sure to specify the UNNUMBERED option of WYLBUR's **RUN** command.

For instance...Use the following WYLBUR command

```
     ?  RUN UNN HOLD NOTIFY
```

2. Interactive Mode

The SAS Display Manager System is an interactive, windowing environment in which actions are performed with a series of commands or function keys. It can be used to prepare and submit SAS programs, view and print the results, and debug and resubmit the programs.
This requires full-screen access to the MVS operating system. The full screen access can be accomplished through TN3270 or NIHnet 3270 Protocol Converter to the MVS System.

To initiate an interactive SAS session on the mainframe you issue the following command at the TSO (READY) prompt

```
ex 'zabcrun.sas.clist(sas)'
```

A Sampling of JCL Statements Used in SAS Programs

This section was put together to answer the most frequently asked questions about getting started with the SAS System and batch processing on the MVS operating system at NIH.

Input Samples

1. **Read raw data when the input data immediately follows a CARDS statement in the SAS program**

```
//  (JOB statement)
//  JCLLIB ORDER=ZABCRUN.PROCLIB
//  EXEC SAS
//SYSIN DD *

DATA TEST;
INPUT variable list ;
CARDS;
(data lines)
;
```

2. **Read raw data on disk and on the MSS in LRECL format**

- **Or with the JCL DD statement**

```
//  (JOB statement)
//  JCLLIB ORDER=ZABCRUN.PROCLIB
//  EXEC SAS
//ddname DD DSN=aaaaiii.dsname,DISP=SHR
//SYSIN DD *

DATA TEST ;
  INFILE ddname ;
  INPUT variable list ;
```

- **Or with the FILENAME statement**

```
//  (JOB statement)
//  JCLLIB ORDER=ZABCRUN.PROCLIB
//  EXEC SAS
//SYSIN DD *

FILENAME filref 'aaaaiii.dsname' ;
DATA TEST;
  INFILE fileref ;
  INPUT variable list ;
```

- **without the FILENAME statement or the JCL DD statement**

```
//  (JOB statement)
//  JCLLIB ORDER=ZABCRUN.PROCLIB
//  EXEC SAS
//SYSIN DD *

DATA TEST;
  INFILE  'aaaaiii.dsname' ;
  INPUT variable list ;
```

3. Read raw data on disk and the MSS in EDIT format when data lines do not extend past column 80

```
//  (JOB statement)
//  JCLLIB ORDER=ZABCRUN.PROCLIB
//  EXEC EDSIN,NAME='aaaaiii.dsname'
//  EXEC SAS
//ddname DD DSN=&INPUT,DISP=(OLD,DELETE)
//SYSIN DD *

DATA TEST;
  INFILE ddname;
  INPUT variable list ;
```

4. Read raw data on disk and the MSS in EDIT format when data lines do extend past column 80

```
//  (JOB statement)
//  JCLLIB ORDER=ZABCRUN.PROCLIB
//  EXEC EDSIN,NAME='aaaaiii.dsname',LRECL=len,BLKSIZE=b
//  EXEC SAS
//ddname DD DSN=&INPUT,DISP=(OLD,DELETE)
//SYSIN DD *

DATA TEST;
  INFILE ddname;
  INPUT variable list ;
```

5. Read raw data on NIH standard tape

```
//  (JOB statement with class B or C)
/*MESSAGE nnnnnn,R
/*ROUTE XEQ TAPE
//  JCLLIB ORDER=ZABCRUN.PROCLIB
//  EXEC SAS
//ddname DD DSN=aaaaiii.dsname,DISP=SHR,
//   UNIT=TAPE,VOL=SER=nnnnnn)
//SYSIN DD *

DATA TEST;
  INFILE ddname;
  INPUT variable list ;
```

- **Or with the FILENAME statement**

```
//  (JOB statement with class B or C)
/*MESSAGE nnnnnn,R
/*ROUTE XEQ TAPE
//  JCLLIB ORDER=ZABCRUN.PROCLIB
//  EXEC SAS
//SYSIN DD *
FILENAME fileref 'aaaaiii.dsname' DISP = SHR
  UNIT = TAPE  VOLSER = 'nnnnnn'   LABEL =(seqnum,lab) ;
DATA TEST;
  INFILE fileref;
  INPUT variable list ;
```

Note:
To access files that are stored on tape under another user's account or initials, include the /*ACCESS statement

/*ACCESS aaaaiii

where 'aaaaiii' is the account/initials combination to which the tape is assigned.

6. Read raw data on 6250 BPI tape

```
//  (JOB statement with class B or C)
/*MESSAGE nnnnnn,R
/*ROUTE XEQ 9TRACKHI
//  JCLLIB ORDER=ZABCRUN.PROCLIB
//  EXEC SAS
//ddname DD DSN=dsname,DISP=SHR,
//  VOL=SER=nnnnnn,UNIT=9TRACKHI,
//  LABEL=(seqnum,lab)
//SYSIN DD *

DATA TEST;
  INFILE ddname;
  INPUT variable list ;
```

- **Or with the FILENAME statement**

```
//  (JOB statement with class B or C)
/*MESSAGE nnnnnn,R
/*ROUTE XEQ 9TRACKHI
//  JCLLIB ORDER=ZABCRUN.PROCLIB
//  EXEC SAS
//SYSIN DD *

FILENAME  filref  'aaaaiii.dsname'  DISP=SHR
  UNIT = '9TRACKHI'  VOLSER = 'nnnnnn'  LABEL = (seqnum,lab)  ;

DATA TEST;
  INFILE fileref ;
  INPUT variable list ;
```

7. Read SAS data set on the disk and the MSS

```
//  (JOB statement)
//  JCLLIB ORDER=ZABCRUN.PROCLIB
//  EXEC SAS
//ddname DD DSN=aaaaiii.dsname,DISP=SHR
//SYSIN DD *

DATA TEST;
  SET ddname.SASname;
```

- **Or with the LIBNAME statement**

```
//  (JOB statement)
//  JCLLIB ORDER=ZABCRUN.PROCLIB
//  EXEC SAS
//SYSIN DD *

LIBNAME libref 'aaaaiii.dsname';

DATA TEST;
  SET libref.SASname;
```

8. Read SAS data set on NIH standard tape

```
//  (JOB statement with class B or C)
/*MESSAGE nnnnnn,R
/*ROUTE XEQ TAPE
//  JCLLIB ORDER=ZABCRUN.PROCLIB
//  EXEC SAS
//ddname DD DSN=dsname,DISP=SHR,
//  VOL=SER=nnnnnn,UNIT=TAPE,
//  LABEL=(seqnum,lab)
//SYSIN DD *

DATA TEST;
  SET ddname.SASname ;
```

• **Or with the LIBNAME statement**

```
//  (JOB statement with class B or C)
/*MESSAGE nnnnnn,R
/*ROUTE XEQ TAPE
//  JCLLIB ORDER=ZABCRUN.PROCLIB
//  EXEC SAS
//SYSIN DD *

DATA TEST;
  SET libref.SASname ;

LIBNAME libref  'aaaaiii.dsname'  UNIT = TAPE  DISP = SHR
  VOLSER = 'nnnnnn'  LABEL = (seqnum,lab) ;
```

9. Read SAS data set on 6250 BPI tape

```
//  (JOB statement with class B or C)
/*MESSAGE nnnnnn,R
/*ROUTE XEQ 9TRACKHI
//  JCLLIB ORDER=ZABCRUN.PROCLIB
//  EXEC SAS
//ddname DD DSN=dsname,DISP=SHR,
//  VOL=SER=nnnnnn,UNIT=9TRACKHI,
//  LABEL=(seqnum,lab)
//SYSIN DD *

DATA TEST;
  SET ddname.SASname ;
```

• **Or with the LIBNAME statement**

```
//  (JOB statement with class B or C)
/*MESSAGE nnnnnn,R
/*ROUTE XEQ 9TRACKHI
//  JCLLIB ORDER=ZABCRUN.PROCLIB
//  EXEC SAS
//SYSIN DD *

DATA TEST;
   SET libref.SASname ;

LIBNAME libref  'aaaaiii.dsname'  UNIT = '9TRACKHI' DISP = SHR
  VOLSER = 'nnnnnn'   LABEL = (seqnum,lab)  ;
```

Output Samples

NOTE:
All FILE, TMP, and MSS data sets are cataloged. All references to data sets on the disk are made through the catalog. One consequence of the all-cataloged environment is that during program development, it is possible to accidentally create multiple copies of permanent files of which only the first copy is cataloged. To avoid this, include an **EXEC DSSCR** statement in the JCL to delete previous copies of the permanent file. **General Form:** // EXEC DSSCR,NAME='aaaaiii.dsname'

10. Write raw data to a file on the disk

```
// (JOB statement)
// JCLLIB ORDER=ZABCRUN.PROCLIB
// EXEC DSSCR,NAME='aaaaiii.dsname'
// EXEC SAS
//ddname DD UNIT=FILE,DISP=(NEW,CATLG),
//  DSN=aaaaiii.dsname,SPACE=(TRK,(s1,s2),RLSE),
//  DCB=(RECFM=FB,LRECL=len,BLKSIZE=b)
//SYSIN DD *

DATA _NULL_;
    /* other-SAS-statements */
FILE ddname;
    /* other-SAS-statements */
```

- **Or with the FILENAME statement**

```
//  (JOB statement)
//  JCLLIB ORDER=ZABCRUN.PROCLIB
//  EXEC DSSCR,NAME='aaaaiii.dsname'
//  EXEC SAS
//SYSIN DD *
FILENAME fileref 'aaaaiii.dsname'  UNIT=FILE  DISP=(NEW,CATLG)
  SPACE=(TRK,(s1,s2),RLSE)  RECFM=FB  LRECL=len  BLKSIZE=b  ;

DATA _NULL_;
    /* other-SAS-statements */
FILE fileref;
    /* other-SAS-statements */
```

NOTE:
Specify DISP=MOD if the data set exists and new records are to be added to the end.

11. Write raw data to a file on the MSS

```
//  (JOB statement)
/*ROUTE XEQ MSS
//  JCLLIB ORDER=ZABCRUN.PROCLIB
//  EXEC DSSCR,NAME='aaaaiii.dsname'
//  EXEC SAS
//ddname DD UNIT=MSS,DISP=(NEW,CATLG),
//  DSN=aaaaiii.dsname,SPACE=(TRK,(s1,s2),RLSE),
//  DCB=(RECFM=FB,LRECL=len,BLKSIZE=b)
//SYSIN DD *

DATA _NULL_;
    /* other-SAS-statements */

FILE ddname;
    /* other-SAS-statements */
```

- **Or with the FILENAME statement**

```
//  (JOB statement)
/*ROUTE XEQ MSS
//  JCLLIB ORDER=ZABCRUN.PROCLIB
//  EXEC DSSCR,NAME='aaaaiii.dsname'
//  EXEC SAS
//SYSIN DD *
FILENAME fileref 'aaaaiii.dsname' UNIT=MSS  DISP=(NEW,CATLG)
SPACE=(TRK,(s1,s2),RLSE)  RECFM=FB  LRECL=len  BLKSIZE=b ;

DATA _NULL_;
    /* other-SAS-statements */
FILE fileref;
    /* other-SAS-statements */
```

12. Write raw data to file on NIH standard tape

```
// (JOB statement with class B or C)
/*MESSAGE nnnnnn,W
/*ROUTE XEQ TAPE
//  JCLLIB ORDER=ZABCRUN.PROCLIB
//  EXEC SAS
//ddname DD UNIT=TAPE,DISP=(NEW,CATLG),DSN=aaaaiii.dsname,
//  VOL=SER=nnnnnn,LABEL=(seqnum,lab),
//  DCB=(RECFM=FB,LRECL=len,BLKSIZE=b)
//SYSIN DD *

DATA _NULL_;
   /* other-SAS-statements */
FILE ddname ;
   /* other-SAS-statements */
```

- **Or with the FILENAME statement**

```
//  (JOB statement with class B or C)
/*MESSAGE nnnnnn,W
/*ROUTE XEQ TAPE
//  JCLLIB ORDER=ZABCRUN.PROCLIB
//  EXEC SAS
//SYSIN DD *

FILENAME  fileref 'aaaaiii.dsname' UNIT = TAPE DISP = (NEW,CATLG)
   VOLSER = 'nnnnnn'   LABEL = (seqnum,lab) RECFM=FB LRECL=len BLKSIZE=b  ;

DATA _NULL_;
     /* other-SAS-statements */
FILE fileref;
   /* other-SAS-statements */
```

13. Write raw data to file on 6250 BPI tape

```
//  (JOB statement with class B or C)
/*MESSAGE nnnnnn,W
/*ROUTE XEQ 9TRACKHI
//  JCLLIB ORDER=ZABCRUN.PROCLIB
//  EXEC SAS
//ddname DD UNIT=9TRACKHI,DISP=(NEW,CATLG),
//  DSN=aaaaiii.dsname,VOL=SER=nnnnnn,
//  LABEL=(seqnum,lab),DCB=(RECFM=FB,LRECL=len,BLKSIZE=b)
//SYSIN DD *

DATA _NULL_;
   /* other-SAS-statements */
FILE ddname;
   /* other-SAS-statements */
```

- **Or with the FILENAME statement**

```
//  (JOB statement with class B or C)
/*MESSAGE nnnnnn,W
/*ROUTE XEQ 9TRACKHI
//  JCLLIB ORDER=ZABCRUN.PROCLIB
//  EXEC SAS
//SYSIN DD *

FILENAME  fileref 'aaaaiii.dsname' UNIT = '9TRACKHI' DISP = (NEW,CATLG)
   VOLSER = 'nnnnnn'   LABEL = (seqnum,lab) RECFM=FB LRECL=len BLKSIZE=b  ;
DATA _NULL_;
   /* other-SAS-statements */
FILE fileref;
   /* other-SAS-statements */
```

14. Write SAS data set to the disk

```
//  (JOB statement)
//  JCLLIB ORDER=ZABCRUN.PROCLIB
//  EXEC DSSCR,NAME='aaaaiii.dsname'
//  EXEC SAS
//ddname DD UNIT=FILE,DISP=(NEW,CATLG),
//  DSN=aaaaiii.dsname,SPACE=(TRK,(s1,s2),RLSE)
//SYSIN DD *

DATA ddname.SASname;
   /* other-SAS-statements */
```

- **Or with the LIBNAME statement**

```
//  (JOB statement)
//  JCLLIB ORDER=ZABCRUN.PROCLIB
//  EXEC DSSCR,NAME='aaaaiii.dsname'
//  EXEC SAS
//SYSIN DD *

LIBNAME libref 'aaaaiii.dsname' UNIT=FILE
   DISP=(NEW,CATLG)  SPACE=(TRK,(s1,s2),RLSE) ;

DATA libref.SASname;
   /* other-SAS-statements */
```

15. Write SAS data set to the MSS

```
                                 //  (JOB statement)
//  JCLLIB ORDER=ZABCRUN.PROCLIB
//  EXEC DSSCR,NAME='aaaaiii.dsname'
//  EXEC SAS
//ddname DD UNIT=MSS,DISP=(NEW,CATLG),
//  DSN=aaaaiii.dsname,SPACE=(TRK,(s1,s2),RLSE)
//SYSIN DD *

DATA ddname.SASname;
   /* other-SAS-statements */
```

- **Or with the LIBNAME statement**

```
//  (JOB statement)
//  JCLLIB ORDER=ZABCRUN.PROCLIB
//  EXEC DSSCR,NAME='aaaaiii.dsname'
//  EXEC SAS
//SYSIN DD *

LIBNAME libref 'aaaaiii.dsname' UNIT=MSS DISP=(NEW,CATLG)
   SPACE=(TRK,(s1,s2),RLSE) ;

DATA libref.SASname;
   /* other-AS-statements */
```

16. Write SAS data set to NIH standard tape

```
//  (JOB statement with class B or C)
/*MESSAGE nnnnnn,W
/*ROUTE XEQ TAPE
//  JCLLIB ORDER=ZABCRUN.PROCLIB
//  EXEC SAS
//ddname DD UNIT=TAPE,DISP=(NEW,CATLG),
//  DSN=aaaaiii.dsname,VOL=SER=nnnnnn,
//  LABEL=(seqnum,lab)
//SYSIN DD *

DATA ddname.SASname;
   /* other-SAS-statements
```

- **Or with the LIBNAME statement**

```
//  (JOB statement with class B or C)
/*MESSAGE nnnnnn,W
/*ROUTE XEQ TAPE
//  JCLLIB ORDER=ZABCRUN.PROCLIB
//  EXEC SAS
//SYSIN DD *

LIBNAME libref  'aaaaiii.dsname'  UNIT = TAPE  DISP = (NEW,CATLG)
  VOLSER = 'nnnnnn'   LABEL = (seqnum,lab)  ;

DATA libref.SASname;
   /* other-SAS-statements */
```

17. Write SAS data set to 6250 BPI tape

```
//  (JOB statement with class B or C)
/*MESSAGE nnnnnn,W
/*ROUTE XEQ 9TRACKHI
//  JCLLIB ORDER=ZABCRUN.PROCLIB
//  EXEC SAS
//ddname DD UNIT=9TRACKHI,DISP=(NEW,CATLG),
//  DSN=aaaaiii.dsname,VOL=SER=nnnnnn,
//  LABEL=(seqnum,lab)
//SYSIN DD *

DATA ddname.SASname;
   /* other-SAS-statements */
```

- **Or with the LIBNAME statement**

```
//  (JOB statement with class B or C)
/*MESSAGE nnnnnn,W
/*ROUTE XEQ 9TRACKHI
//  JCLLIB ORDER=ZABCRUN.PROCLIB
//  EXEC SAS
//SYSIN DD *
LIBNAME libref  'aaaaiii.dsname'  UNIT = '9TRACKHI'  DISP = (NEW,CATLG)
   VOLSER = 'nnnnnn'   LABEL = (seqnum,lab)  ;

DATA libref.SASname;
   /* other-SAS-statements */
```

18. Sample program with both input and output.

Suppose there are two sources of input
1) Read a raw data file on the MSS in EDIT format with records extending past column 80
2) read a SAS data set on FILE
3) Write a SAS data set to an NIH standard tape.

```
//  (JOB statement with class B or C)
/*MESSAGE nnnnnn,W
/*ROUTE XEQ TAPE
//  JCLLIB ORDER=ZABCRUN.PROCLIB
//  EXEC EDSIN,NAME='aaaaiii.mssdsname',LRECL=len,BLKSIZE=b
//  EXEC SAS
//ddname1 DD DSN=&INPUT,DISP=(OLD,DELETE)
//ddname2 DD DSN=aaaaiii.diskdsname,DISP=SHR
//ddname3 DD UNIT=TAPE,DISP=(NEW,CATLG),
//  DSN=aaaaiii.tapedsname,VOL=SER=nnnnnn,
//  LABEL=(seqnum,lab)
//SYSIN DD *
```

Access SPSS Files

The SPSS read-only engine enables you to access files that were created with SPSS software as if these files are written by the SAS System:

- SPSS engine accesses standard SPSS files created under Release 9 of SPSS as well as SPSS-X system files and portable files. The SPSS portable files can originate from any operating system. The engine automatically determines which type of SPSS file it is reading and reads the file accordingly.

Assigning a Libref to an SPSS File

In order to access a standard SPSS file you include a LIBNAME statement to assign a *libref* to the SPSS file. Specify the SPSS option in the LIBNAME statement. This option is called an engine and tells SAS that the file is an SPSS file.

SPSS files do not have an internal member name as in SAS. Consequently you use the word *_FIRST_* in your SAS programs to refer to the data set name.

Example 1: Reads a portable SPSS file in a DATA step

```
//  (JOB statement)
//  JCLLIB ORDER=ZABCRUN.PROCLIB
//  EXEC SAS
//SYSIN DD *

LIBNAME libref  SPSS  'aaaaiii.dsname' ;

DATA TEST ;
   SET libref._FIRST_ ;
```

Example 2: Processes a standard SPSS file with the PROC CONTENTS procedure

```
//  (JOB statement)
//  JCLLIB ORDER=ZABCRUN.PROCLIB
//  EXEC SAS
//SYSIN DD *

LIBNAME libref  SPSS  'aaaaiii.dsname' ;

PROC CONTENTS data = libref._FIRST_ ;
```

Access BMDP Files

The read-only BMDP engine enables you to access files that were created with BMDP software as if these files are written by the SAS System:

Assigning a Libref to a BMDP FIle

In order to access a BMDP file, you must use the LIBNAME statement to assign a *libref* to the file. Specify the BMDP engine in the LIBNAME statement as shown in the following examples.

Because there can be multiple "save" files in a single physical BMDP file, you use the value of the BMDP CODE=argument as the name of the SAS data file. For example, if the BMDP 'save" file contains CODE=abc, and if the *libref* is xxx, you reference the file as xxx.abc.

```
     LIBNAME xxx  BMDP  'aaaaiii.dsname'; /*reference the physical BMDP file */

        DATA test ;
          SET xxx.abc ;
```

Example 1:

In your SAS program, if you want to access the first BMDP "save" file in the operating system data set, or if there is only one "save" file, you can refer to the file as _FIRST_. This approach is convenient if you do not know the BMDP CODE=argument as the name of the SAS data file.

```
     LIBNAME xxx  BMDP  'aaaaiii.dsname' ; /*reference the physical BMDP file */

        DATA test ;
          SET xxx._FIRST_ ;
```

Example 2:

The following statements assign a *libref* to the data set and then run a PROC CONTENTS and a PROC PRINT on the BMDP file.

```
     LIBNAME xxx  BMDP  'aaaaiii.dsname' ; /*reference the physical BMDP file*/

        PROC CONTENTS data = xxx.abc;

        PROC PRINT data = xxx.abc ;
```

Access DB2 tables from SAS

You can access **DB2** tables using SAS in batch mode or in full-screen mode (e.g. TN3270) after setting up your sessions appropriately.

There are two methods of accessing DB2 data:

1. Use the **SQL Pass-Through Facility**
   Allows you to access DB2 data using SQL syntax with PROC SQL.

2. Use **Access and View Descriptors**
   Allows you to create views of the DB2 data so you can access it as if it were a SAS data set.

The following documentation is available at the
CIT Technical Information Office (301-594-3278).

1. SAS/ACCESS Software for Relational Databases and the DB2 chapter
2. SAS/ACCESS Software, Changes and Enhancements, SQL Procedure
   Pass-Through Facility

I. Setup

A. Batch Mode

To use DB2 data from SAS:
Specify the subsystem you want to access and include one of the following JCL statements. The **ROUTE** statement should be placed immediately after the **JOB** statement.

Subsystem	ROUTE Statement
DSND	/*ROUTE XEQ DB2DEV
DSNP	/*ROUTE XEQ DB2PROD

To invoke SAS use the appropriate **EXEC** statement below:

Subsystem	EXEC Statement
DSNP	// EXEC SASDSNP
DSND	// EXEC SASDSND

In the SAS program include an **OPTIONS** statement specifying the subsystem to which you are connecting:

Subsystem	OPTIONS Statement
DSND	options db2ssid=dsnd;
DSNP	options db2ssid=dsnp;

B. Full-Screen Mode

To access DB2 data from SAS in full-screen mode you must first create a user profile. This file will be invoked when you invoke a 3270 session. The profile should contain the command:

```
  DBALLOC SYSP('''ZABCRUN.SAS.CLIST''')
```

The profile name and subsystem should be specified:

Subsystem	**Profile**
DSND	aaaaiii.@DB.DEV.PROFILE.CLIST
DSNP	aaaaiii.@DB.PROD.PROFILE.CLIST

Log on to DB2 **not** SAS. To log on use one of the commands below depending on what DB2 subsystem you want to access:

Subsystem	**DB Command**
DSND	DB aaaaiii DBDEV
DSNP	DB aaaaiii DBPROD

After the READY prompt, invoke SAS by specifying the appropriate subsystem and corresponding command

Subsystem	**Command**
DSND	SASDSND
DSNP	SASDSNP

II. Using the SQL Pass-Through Facility

A. Batch Mode

1. Querying DB2 data

The SQL Pass-Through Facility is part of Base SAS and allows you to use SQL within SAS. To use SQL in SAS use the SQL procedure. To access DB2 data you must include the **CONNECT TO DB2** statement and the **SELECT * FROM CONNECTION TO DB2** statement as shown in the example below. In a separate **SELECT** statement and enclosed in parenthesis you specify the table, columns, rows and statistics to select.

The following program queries the DB2 table CUSTOMERS. Notice that it invokes the DB2 subsystem **DSNP**. If your data is in subsystem **DSND** you would need to modify the **JCL** and the **SSID=** option in the **CONNECT** statement as stated above.

```
  //iii JOB (aaaa,box,class),lastname
  /*ROUTE XEQ DB2PROD
  //  JCLLIB ORDER=ZABCRUN.PROCLIB
  //  EXEC SASDSNP
  //SYSIN DD *

  proc sql;
   connect to db2 (ssid=dsnp);
   select * from connection to db2
    (select name, contact from aaaaiii.customers);
   disconnect from db2;
  quit;
```

2. Creating a SAS data set

To create a SAS data set use the **CREATE TABLE** statement as shown in the example below. The program creates a temporary SAS data set called CUSTOMER. To create a permanent SAS data set specify a two-part SAS data set name and a **LIBNAME** statement.

```
  //iii JOB (aaaa,box,class),lastname
  /*ROUTE XEQ DB2PROD
  //  JCLLIB ORDER=ZABCRUN.PROCLIB
  //  EXEC SASDSNP
  //SYSIN DD *

  proc sql;
   connect to db2 (ssid=dsnp);
   create table customer as
   select * from connection to db2
    (select name, contact from aaaaiii.customers);
   disconnect from db2;
  quit;
```

B. Full-Screen Mode

1. Querying DB2 data

The SQL Pass-Through Facility is part of Base SAS and allows you to use **SQL** within SAS. To use SQL in SAS use the **SQL** procedure. To access DB2 data you must include the **CONNECT TO DB2** statement and the **SELECT * FROM CONNECTION TO DB2** statement as shown in the example below. In a separate **SELECT** statement and enclosed in parenthesis you specify the table, columns, rows and statistics to select.

This program queries the **DB2** table CUSTOMERS. Notice that we are invoking the **DB2** subsystem **DSNP**. If your data is in subsystem **DSND** you would need to modify the JCL and the **SSID=** option in the **CONNECT** statement.

```
  proc sql;
   connect to db2 (ssid=dsnp);
   select * from connection to db2
    (select name, contact from aaaaiii.customers);
   disconnect from db2;
  quit;
```

2. Creating a SAS data set

To create a SAS data set use the **CREATE TABLE** statement as shown in the example below. The program creates a temporary SAS data set called CUSTOMER. To create a permanent SAS data set specify a two-part SAS data set name and a **LIBNAME** statement.

```
  proc sql;
   connect to db2 (ssid=dsnp);
   create table customer as
   select * from connection to db2
    (select name, contact from aaaaiii.customers);
   disconnect from db2;
  quit;
```

III. Access and View Descriptors

To access DB2 data as if it were a SAS data set you must:

1. create an access descriptor using **PROC ACCESS**
2. create a view descriptor using **PROC ACCESS**
3. use the view descriptor as you would use a standard SAS data set

Below are some examples of creating the descriptors. These are very simple examples. You should read the DB2 Chapter of the guide **"SAS/ACCESS Software for Relational Databases"** for the complete syntax of the ACCESS procedure.

A. Batch Mode

1. Creating access and view descriptors

As mentioned before in order to access **DB2** data from SAS you need to create an access descriptor and a view descriptor. An access descriptor holds information about the **DB2** table such as column names, data types, SAS variable names and formats. A view descriptor defines some or all of the columns and rows described by the access descriptor.
You may create more than one of each if needed. A view descriptor is considered a SAS data set and is used as you would any SAS data set. To create access and view descriptors use the **ACCESS** procedure.

In the example below we first use a **LIBNAME** statement to create a SAS library. Then the ACCESS procedure creates the access descriptor called CUSTOMER and a view descriptor called CUSTOMER. The name of the access descriptor and the view descriptor can be the same. SAS recognizes one from the other from the type of descriptor it is: **ACCESS** or ***VIEW***. You only need to specify the type when you create them (i.e. in the **CREATE** statement).

```
//iii JOB (aaaa,box,class),lastname
/*ROUTE XEQ DB2PROD
//  JCLLIB ORDER=ZABCRUN.PROCLIB
//  EXEC SASDSNP
//SYSIN DD *

options db2ssid=dsnp;

libname out 'aaaaiii.sasdb2.library' unit=file
            disp=(new,catlg) space=(trk,(1,20));

proc access dbms=db2;
  create out.customer.access;
    table=aaaaiii.customers;
    assign=yes;
  create out.customer.view;
    select customer name contact;
```

2. Using a view descriptor

The following program accesses the view descriptor CUSTOMER that was created in the previous example. Even though it points to a **DB2** table you can use it as if it were a SAS data set.

```
//iii JOB (aaaa,box,class),lastname
/*ROUTE XEQ DB2PROD
//  JCLLIB ORDER=ZABCRUN.PROCLIB
//  EXEC SASDSNP
//SYSIN DD *

options db2ssid=dsnp;

libname in ’aaaaiii.sasdb2.library’ disp=shr;

proc means data=in.customer;
 var sales;
run;

data md;
 set in.customer;
 if state=’MD’ then sales=sales*1.05;
run;

proc print;
run;
```

B. Full-Screen Mode

1. Creating access and view descriptors

```
libname out ’aaaaiii.sasdb2.library’ unit=file disp=(new,catlg)
            space=(trk,(1,20));
proc access accdesc=out.customer function=c;
run;
```

2. Using a view descriptor

```
options db2ssid=dsnp;

libname in ’aaaaiii.sasdb2.library’ disp=shr;

proc means data=in.customer;
 var sales;
run;

data md;
 set in.customer;
 if state=’MD’ then sales=sales*1.05;
run;

proc print;
run;
```

PROC BMDP Procedure

Objective: **Calls a BMDP program to analyze data in a SAS data set**

or

Converts a SAS data set to a BMDP "save" file

Syntax:

```
PROC BMDP options  ; /* reference the BMDP program you want to invoke*/
VAR variable-list  ; /*optional */ /*when you do not include a VAR statement, the BMDP
                     program uses all the numeric variables in the SAS data set. */
BY variable-list ;   /* optional */
PARMCARDS;           /* indicates the BMDP control language follows */
BMDP control statements
.
;
```

The following options can be used in the PROC BMDP statement:

CODE = *save-file*
assigns a name to the BMDP "save" file that the BMDP procedure creates from a SAS data set. The *save-file* corresponds to the CODE sentence in the BMDP INPUT paragraph. For example, you can use the following statement:

```
PROC BMDP PROG=BMDP3S  CODE=JUDGES;
```

Then, the BMDP INPUT paragraph must contain the following sentence:

```
CODE = 'JUDGES'
```

CODE = usually is not necessary in the PROC BMDP statement. When CODE= is not specified, the name of the BMDP "save" file is the SAS data set name.

If you are converting a SAS data set to a BMDP "save" file, include the CODE sentence in the BMDP SAVE paragraph to name the BDMP "save" file. To use the name of the SAS data set, specify that name of the SAS data set in the CODE sentence of the BMDP INPUT paragraph.

PROG=BMDP*nn* specifies the BMDP program that you want to run.
Note: If you want only to convert a SAS data set to a BMDP "save' file and do not want to run a BMDP program, omit the PROG = option and include the UNIT = option

UNIT=*n*

specifies the FORTRAN logical unit number for the BMDP "save" file that the BMDP procedure creates. The value you specify for *n* must correspond to the UNIT= value that is specified in the INPUT paragraph of the BMDP control language.
If you omit this option, *n* defaults to 3 and FT03F001 is used as the *fileref* for the "save" file. The following message is also printed:

Note: The UNIT= option was not specified. Unlike Version 5, the UNIT = option is required. Therefore, UNIT=3 is assumed. Ensure that the INPUT paragraph uses a unit of 3, or explicitly specify the correct UNIT value.

WORKSPCE = *nn* or PARM = *nn*

Controls the allocation of a workspace in BMDP. The WRKSPCE = or PARM = value is passed as a parameter to BMDP programs and corresponds to the WRKSPCE = feature in BMDP MVS cataloged procedures. The default value for ***nn*** is 30. If ***nn*** is less than 100, then its value represents kilobytes. If it is greater than 100, then its value represents bytes.

PARMCARDS Statement

PARMCARDS:

The PARMCARDS statement indicates that the BMDP control language follows.

BMDP Control Statements

Put your BMDP control language statements after the PARMCARDS statement. These statements are similar for all BMDP programs.

The BMDP INPUT paragraph must include UNIT and CODE sentences. The value of these sentences must match the UNIT= and the CODE= values that are given in the PROC BMDP statement. (If the PROC BMDP statement does not specify a UNIT= value, then use 3 as the UNIT= value in the BMDP statements.) Use the SAS data set name as the CODE value unless you have used the CODE= option in the PROC BMDP statement to specify a different name. Omit the VARIABLES paragraph from the BMDP statements, because it is not needed when your input is a "save’ file.

How Missing Values Are Handled

Before the BMDP procedure sends data to BMDP, it converts missing SAS values to the standard BMDP missing value. When you use the NOMISS option in the PROC BMDP statement, observations that contain missing values are excluded from the data set that is sent to the BMDP program.

NOTE:
In order to use the BMDP procedure in a SAS session use the following JCL statement to request the cataloged procedure **SASBMDP** rather than the usual cataloged procedure **SAS**.

```
//  EXEC SASBMDP
```

Example 1:

Calls a BMDP program to analyze data in a SAS data set

```
                                                    //  (JOB statement)
//  JCLLIB ORDER=ZABCRUN.PROCLIB
//  EXEC SASBMDP
//ddname DD DSN=aaaaiii.dsname,DISP=SHR
//SYSIN DD *

PROC BMDP DATA = ddname.sasdsname UNIT=3 PROG=BMDP1D ;
VAR variables ;
PARMCARDS ;
/PROB TITLE IS ’SAS/BMDP’.
/INPUT UNIT = 3.  CODE = ’sasdsname’.
/END
/FINISH
;
```

Example 2:

PROC BMDP calls the BMDP program BMDP1D to analyze the SAS data set and the result is stored in a BMDP "save' file.

```
// (JOB statement)
// JCLLIB ORDER=ZABCRUN.PROCLIB
// EXEC DSSCR,NAME='aaaaiii.dsname2'
// EXEC SASBMDP
//ddname DD DSN=aaaaiii.dsname1,DISP=SHR
//FTnnF001 DD UNIT=FILE,DISP=(NEW,CATLG),
// DCB=(RECFM=VBS,LRECL=6352,BLKSIZE=6356),
// SPACE=(TRK,(s1,s2)),DSN=aaaaiii.dsname2
//SYSIN DD *

PROC BMDP DATA = ddname.sasdsname UNIT=3 PROG=BMDP1D ;
VAR variables ;
PARMCARDS ;
/PROB TITLE = 'SAS/BMDP'.
/INPUT UNIT = 3.  CODE = 'sasdsname'.
/SAVE  Code = 'bmdpname'.  NEW.  UNIT=nn.
/END
/FINISH
;
```

where 'nn,' in the UNIT=nn sentence specifies the logical unit number for the BMDP 'save" file. It is an integer between 08 and 15.
UNIT = nn in the SAVE paragraph should refer to the FTnnF001 fileref in the JCL DD statement.

The word NEW must be in the SAVE paragraph.

Note the BMDP program statements UNIT=3. and CODE = 'sasdsname'. The results are stored in the BMDP "save" file NEW.

When using BMDP from SAS, do not use the VARIABLES paragraph in BMDP. The INPUT paragraph with the UNIT and CODE sentences is required

Example 3:

Assign a libref to a BMDP file.

```
LIBNAME  FT04F001  BMDP  ;

Data  _null_  ;
SET  FT04F001.bmdpname ;
PUT  ALL ;
```

The LIBNAME statement associates the libref FT04F001 with the BMDP engine so that SAS knows which engine to use to access the data.

The DATA Step reads the BMDP "save' file bmdpname.

PROC CONVERT Procedure

Objective: Converts BMDP and SPSS system files to SAS data sets

PROC CONVERT produces one output SAS data set, but no printed output. The new SAS data set contains the same information as the input system file; exceptions are noted below.

You can create a temporary or a permanent SAS data set. To create a temporary data set specify a one-part name after OUT = option. Otherwise, include a LIBNAME statement and include the libref after the OUT = option together with a SAS data set.

How Missing Values Are Handled

If a numeric variable in the input data set has no value or has a system missing value, PROC CONVERT assigns a missing value to it.
Caution………
Because some translation of variable names can occur, ensure that the translated names will be unique.

How Variable Names from the SPSS "save" file are used in the SAS data set.

- SPSS variable names and labels become variable names and labels without change.
- SPSS alphabetic variables become SAS character variables of length 4.
- SPSS blank values are converted to SAS missing values.
- SPSS print formats become SAS formats
- SPSS default precision of no decimal places becomes part of the variables formats.
- The SPSS DOCUMENT data are copied so that the CONTENTS procedure can display them.
- SPSS value labels are copied.

Example: Convert a standard SPSS file to a permanent SAS data set.
(does not apply to version 5 SAS data sets))

```
//  (JOB statement)
//  JCLLIB ORDER=ZABCRUN.PROCLIB
//  EXEC SAS
//IN1   DD DSN='aaaaiii.dsname' DISP=SHR
//SYSIN DD *

LIBNAME   IN1  SPSS  ;
LIBNAME  OUT1  'aaaaiii.dsname'  UNIT=FILE
 DISP=(NEW,CATLG)  SPACE=(TRK,(s1,s2),RLSE)  ;

PROC CONVERT   SPSS = IN1   OUT = OUT1 ;
```

How variable names from the BMDP "save" file are used in the SAS data set

Variable names from the BMDP "save" file are used in the SAS data set, except that non-trailing blanks and all special characters are converted to underscores in the SAS variable names. The subscript in BMDP variable names, such as x(1), becomes part of the SAS variable name, with the parentheses omitted: X1. Alphabetic BMDP variables become SAS character variables of length 4. Category records from BMDP are not accepted.

Example: Convert a BMDP "save" file to a permanent SAS data set.

```
//  (JOB statement)
//  JCLLIB ORDER=ZABCRUN.PROCLIB
//  EXEC SAS
//IN1   DD DSN='aaaaiii.bmdp.savefile' DISP=SHR
//SYSIN DD *
LIBNAME  IN1  BMDP  ;
LIBNAME  OUT1  'aaaaiii.dsname'  UNIT=FILE  DISP=(NEW,CATLG)
  SPACE=(TRK,(s1,s2),RLSE)  ;
PROC CONVERT   BMDP = IN1    OUT = OUT1  ;
```

Transporting SAS Files Between Operating Systems

Introduction:

Different computer architectures typically have binary incompatibilities such as different representations for floating-point numbers or different character-encoding standards. Therefore, if you need to move SAS files to machines that are running different operating systems, you usually must convert those files to and from neutral, intermediary format called *transport format*. The only exception is that transport format is not required when moving *tape-format* libraries between MVS and CMS.

A convert file is called a *transport file*, and the process of moving a file from one host operating system to another is called *transporting*. Therefore, a transport file can be moved to any release of the SAS System on any host.

NOTE:
The following types of SAS files are host-dependent and therefore cannot be transported:

a) Indexes
b) Views
c) Access descriptors
d) Stored programs

The SAS System provides three ways to transport files:

1) XPORT engine with the PROC COPY procedure
2) CPORT and CIMPORT procedures
Use the CPORT and CIMPORT procedures to transport SAS catalogs. You can use these procedures to transport most types of SAS catalog entries except for compiled macros and IML modules. These files must be created again after their associated data sets have been installed on the receiving host.

The CPORT/CIMPORT transport format is different from the transport format that is created by the XPORT engine. The CPORT/CIMPORT transport format is recognized by the release on which it was produced or by *later* releases. Because of enhancements that have been made to catalog structures, you *sometimes* cannot move catalogs to earlier releases of the SAS System under MVS. For example, if you use PROC CPORT to create a transport file in the 6.09 Enhanced release, you cannot import that file in Release 6.07. However, from hosts that are running Release 6.11 or later of the SAS System, you can use the V608 option with PROC CPORT to convert most 6.11 catalog entry types to a transport file that can then be imported by PROC CIMPORT on 6.08 or 6.09 hosts.

3) Separately licensed SAS/CONNECT product.
For more information about the SAS/CONNECT software product, see the documentation *SAS/CONNECT Software: Usage and Reference.*

CAUTION!
If you attempt to use file transfer software (other than SAS/CONNECT software) to copy SAS files across operating systems without first converting them to transport files by using the PROC COPY or the PROC CPORT procedure, the files become corrupted and therefore, unreadable.

Moving Files via tapes

Transport format is not required when you are moving tape-format libraries between MVS and CMS. But for moving files via tape between other hosts, observe the following:

1) The tape must be nonlabeled and be recorded at a density that can be read on the destination machine. Because tape labels are processed differently on different hosts, reading a file from a standard labeled tape may be somewhat complicated on the receiving host.

2) Only one transport library should be written per tape; however, the transport library can contain as many members as you want.

3) To ensure that there is never any question about which DCB characteristics have been specified, SAS Institute recommends that you also use the DCB characteristics.

```
//  DCB=(RECFM=FB,LRECL=80,BLKSIZE=8000)
```

4) You may encounter problems during importing if your transport file spans more than one tape. (Again, complications may arise because of differences in how different hosts process multivolume tapes.) Therefore, instead of using multivolume tapes, split the original library into two or more librarires and create a separate, unlabeled tape for each one. You can combine the resulting libraries on the receiving host.

General Steps:

Here is the general procedure for transporting a SAS file from one host to another:

1. On the sending host, convert the file to transport format.
2. Move the transport file to the receiving host via nonlabeled tape, diskette, or communication software(such as FTP).
3. On the receiving host, import the transport file. That is, convert the transport to the file format that is used on the receiving host.

A transport file created by PROC COPY cannot be read by PROC CIMPORT and vice versa.

FTP can be used to move a SAS transport file between the PC and the mainframe. It is necessary to specify that a binary file is being moved and then select from the Menu bar…

FTP → COMMAND → Send Quoted Command to Remote →
Quoted Command:

```
Site  RECFM(FB)  LRECL(80)  BLKSIZE(8000)
```

A Sampling of Programs to Create and to Import SAS Transport Files

1. Create a SAS transport file on 6250 BPI nonlabeled tape using PROC COPY with the XPORT engine

```
//  (JOB statement with class B or C)
/*ROUTE XEQ 9TRACKHI
/*MESSAGE nnnnnn,WS
//  JCLLIB ORDER=ZABCRUN.PROCLIB
//  EXEC SAS
//ddname1 DD DSN=aaaaiii.dsname1,DISP=SHR
//ddname2 DD UNIT=9TRACKHI,DISP=(NEW,CATLG),
//  DSN=aaaaiii.dsname2,VOL=SER=nnnnnn,
//  DCB=(RECFM=FB,LRECL=80,BLKSIZE=8000),LABEL=(seqnum,NL)
//SYSIN DD *

LIBNAME ddname2 XPORT ;
PROC COPY IN=ddname1 OUT=ddname2 ;
```

Note: To convert only selected SAS data sets in the library to transport format, use PROC COPY procedure with the SELECT statement.

2. Import a SAS transport file from 6250 BPI nonlabeled tape using PROC COPY with the XPORT engine

```
//  (JOB statement with class B or C)
/*ROUTE XEQ 9TRACKHI
/*MESSAGE nnnnnn,RS
//  JCLLIB ORDER=ZABCRUN.PROCLIB
//  EXEC DSSCR,NAME='aaaaiii.dsname2'
//  EXEC SAS
//ddname1 DD UNIT=9TRACKHI,DISP=SHR,
//  DSN=aaaaiii.dsname1,VOL=SER=nnnnnn,
//  DCB=(RECFM=FB,LRECL=80,BLKSIZE=8000),LABEL=(seqnum,NL)
//ddname2 DD UNIT=FILE,DISP=(NEW,CATLG),
//  DSN=aaaaiii.dsname2,SPACE=(TRK,(s1,s2))
//SYSIN DD *

LIBNAME ddname1 XPORT ;
PROC COPY IN=ddname1 OUT=ddname2 ;
```

3. Create a SAS transport file on 6250 BPI nonlabeled tape using PROC CPORT

```
//  (JOB statement with class B or C)
/*ROUTE XEQ 9TRACKHI
/*MESSAGE nnnnnn,WS
//  JCLLIB ORDER=ZABCRUN.PROCLIB
//  EXEC SAS
//ddname DD DSN=aaaaiii.dsname1,DISP=SHR
//SASCAT DD UNIT=9TRACKHI,DISP=(NEW.CATLG),
//  DSN=aaaaiii.dsname2,VOL=SER=nnnnnn,
//  DCB=(RECFM=FB,LRECL=80,BLKSIZE=8000),LABEL=(seqnum,NL)
//SYSIN DD *

PROC CPORT LIBRARY=ddname TAPE ;
```

4. Import a SAS transport file from 6250 BPI nonlabeled tape using PROC CIMPORT

```
//  (JOB statement with class B or C)
/*ROUTE XEQ 9TRACKHI
/*MESSAGE nnnnnn,RS
//  JCLLIB ORDER=ZABCRUN.PROCLIB
//  EXEC DSSCR,NAME='aaaaiii.dsname2'
//  EXEC SAS
//SASCAT DD UNIT=9TRACKHI,DISP=SHR,
//  DSN=aaaaiii.dsname1,VOL=SER=nnnnnn,
//  DCB=(RECFM=FB,LRECL=80,BLKSIZE=8000),LABEL=(seqnum,NL)
//ddname DD UNIT=FILE,DISP=(NEW,CATLG),
//  DSN=aaaaiii.dsname2,SPACE=(TRK,(s1,s2))
//SYSIN DD *

PROC CIMPORT LIBRARY=ddname TAPE ;
```

5. Create a SAS transport file to the disk using PROC CPORT

```
//iii JOB (aaaa,,class),name
//   JCLLIB ORDER=ZABCRUN.PROCLIB
//  EXEC SAS
//STD1 DD DSN=aaaaiii.v6sas.formats,DISP=SHR
//TRANFILE DD DSN=aaaaiii.transport.file,DISP=(NEW,CATLG),
//  UNIT=FILE,DCB=(RECFM=FB,LRECL=80,BLKSIZE=8000),
//  SPACE=(TRK,(s1,s2))
//SYSIN DD *
```

Or use the LIBNAME and FILENAME statements rather than the JCL (DD statements).

```
LIBNAME STD1 'aaaaiii.v6sas.formats';

FILENAME TRANFILE 'aaaaiii.transport.file'
  UNIT=FILE DISP=(NEW,CATLG) SPACE=(TRK,(s1,s2))
  RECFM=FB LRECL=80 BLKSIZE=8000 ;

PROC CPORT LIBRARY=STD1 FILE=TRANFILE ;
```

6. Import a SAS transport file to the disk using PROC CIMPORT

```
//iii JOB (aaaa,,class),name
//   JCLLIB ORDER=ZABCRUN.PROCLIB
//  EXEC SAS
//TRANFILE DD DSN=aaaaiii.transport.file,DISP=SHR,
//         DCB=(RECFM=FB,LRECL=80,BLKSIZE=8000)
//OUT1  DD DSN=aaaaiii.v6sas.formats,DISP=(NEW,CATLG),
//      UNIT=FILE,SPACE=(TRK,(s1,s2))
//SYSIN DD *
```

Or use LIBNAME and FILENAME statements rather than the JCL (DD statements)

```
FILENAME TRANFILE 'aaaaiii.transport.file'
 RECFM=FB LRECL=80 BLKSIZE=8000 DISP=SHR ;

LIBNAME OUT1 'aaaaiii.formats' UNIT=FILE
 DISP=(NEW,CATLG)   SPACE=(TRK,(s1,s2)) ;

PROC CIMPORT LIBRARY = OUT1 INFILE = TRANFILE ;
```

7. Using the XPORT Engine with the SAS DATA Step to create a transport file

You can also convert a SAS data set to transport format by using the XPORT engine in conjunction with the SAS DATA step. In this method, you specify the XPORT engine on the LIBNAME statement for the transport file. Then use the SAS DATA step to create the file as you would a SAS data set. The following example creates a data set in transport format.

```
//iii JOB (aaaa,,class),name
//   JCLLIB ORDER=ZABCRUN.PROCLIB
//  EXEC SAS
//STD1 DD DSN=aaaaiii.dsname1,DISP=SHR
//TRANS  DD DSN=aaaaiii.dsname2,DISP=(NEW,CATLG),
//  UNIT=FILE,DCB=(RECFM=FB,LRECL=80,BLKSIZE=8000),
//  SPACE=(TRK,(s1,s2))
//SYSIN DD *

LIBNAME trans   XPORT ;

DATA trans.xprtest  ;
  SET std1.test1  ;
     /* other-SAS-statements */
```

One disadvantage to this method is that you can create only one data set at a time. Also, you must specify the name of a particular data set in your program.

Note:
In addition to PROC COPY and the SAS DATA step, you can use the XPORT engine in conjunction with any SAS procedure that reads or writes SAS data sets. Simply specify the XPORT option in the LIBNAME statement before using the libref in the PROC statement.

A Sampling of Programs to Transfer SAS Transport Files Between Operating Systems

1. Download SAS data set from MVS to PC-environment

Step 1. Create a SAS transport file to the disk on MVS.

```
//iii JOB (aaaa,,class),name
//   JCLLIB ORDER=ZABCRUN.PROCLIB
//  EXEC SAS
//STD1 DD DSN=aaaaiii.dsname1,DISP=SHR
//TRANS  DD DSN=aaaaiii.dsname2,DISP=(NEW,CATLG),
//  UNIT=FILE,DCB=(RECFM=FB,LRECL=80,BLKSIZE=8000),
//  SPACE=(TRK,(s1,s2))
//SYSIN DD *

LIBNAME TRANS  XPORT ;

PROC COPY IN = STD1  OUT = TRANS ;
```

Step 2: Transfer the SAS transport file

a) if you transfer the file by way of a software communication package(like KERMIT, ProComm or other) to the PC environment be certain to specify the following parameters...

```
SET DATA BINARY
SET RECFM FB
SET LRECL 80
SET BLKSIZE 8000
```

b) if you transfer the SAS transport file via FTP and you are using WIN95 or WIN NT then
- Select **Binary** under the **MODE** menu(located in the center of the FTP windows)
- Select **"send quoted command to remote"** from the COMMAND menu in the FTP window and enter the following command:

```
SITE RECFM(FB) LRECL(80) BLKSIZE(8000)
```

Step 3: Re-convert(import) the SAS transport file to a SAS data set on the PC

After transferring the SAS transport file, you may use the following sample on the PC to re-convert the SAS file to a SAS data set.

```
LIBNAME TRANS  XPORT  'C:\mydir\filename'  ;

LIBNAME  OUT1   'C:\mydir'  ;

PROC COPY  IN = TRANS  OUT=OUT1 ;
RUN;
```

2. Download SAS Library with member-type CATALOG from MVS to PC-environment

Please Note:
If the SAS Library contains member-type CATALOG then you must use PROC CPORT to create a SAS transport file.

Step 1: Create a SAS transport file to the disk on MVS.

```
//iii JOB (aaaa,,E),name
//   JCLLIB ORDER=ZABCRUN.PROCLIB
//  EXEC SAS
//STD1 DD DSN=aaaaiii.v6sas.formats,DISP=SHR
//TRANFILE DD DSN=aaaaiii.transport.file,DISP=(NEW,CATLG),
//  UNIT=FILE,DCB=(RECFM=FB,LRECL=80,BLKSIZE=8000),
//  SPACE=(TRK,(s1,s2))
//SYSIN DD *
```

Or use the LIBNAME and FILENAME statements rather than JCL (DD statements).

```
LIBNAME  STD1  'aaaaiiii.v6sas.formats'  ;
FILENAME  TRANSFILE 'aaaaiii.transport.file'
  UNIT=FILE   DISP=(NEW,CATLG)   SPACE=(TRK,(s1,s2))
  RECFM=FB  LRECL=80  BLKSIZE=8000  ;

PROC CPORT LIBRARY=STD1 FILE=TRANFILE ;
```

Step 2: Transfer the SAS transport file

a) if you transfer the file by way of a software communication package(like KERMIT, ProComm or other) to the PC environment be certain to specify the following parameters...

```
SET DATA BINARY
SET RECFM FB
SET LRECL 80
SET BLKSIZE 8000
```

b) if you transfer the SAS transport file via FTP and you are using WIN95 or WIN NT then
- Select **Binary** under the **MODE** menu(located in the center of the FTP WINDOW)
- Select **"send quoted command to remote"** from the COMMAND menu in the FTP window and enter the following command:
  SITE RECFM(FB) LRECL(80) BLKSIZE(8000)

Step 3: Re-convert(import) the SAS transport file to a SAS data set on the PC

After transferring the SAS transport file, you may use the following sample on the PC to re-convert the SAS transport file to a SAS data set.

```
FILENAME TRANFILE  'C:\mydir\filename';
LIBNAME  OUT1  'C:\mydir'  ;
PROC CIMPORT LIBRARY = OUT1 INFILE = TRANFILE ;
RUN;
```

3. Upload SAS data sets from PC-environment to MVS

Step 1: Create a SAS transport file on the PC environment.

```
LIBNAME STD1     'C:\mydir' ;

LIBNAME TRANS   XPORT  'C:\mydir\filename' ;

PROC COPY  IN = STD1  OUT=TRANS ;
   Select membername ;
RUN;
```

Step 2: Transfer the SAS transport file

a) if you transfer the file by way of a software communication package(like KERMIT, ProComm or other) to the MVS environment be certain to specify the following parameters...

```
SET DATA BINARY
SET RECFM FB
SET LRECL 80
SET BLKSIZE 8000
```

b) if you transfer the SAS transport file via FTP and you are using WIN95 or WIN NT then
- Select **Binary** under the MODE menu(located in the center of the FTP WINDOW)
- Select **"send quoted command to remote"** from the COMMAND menu in the FTP window and enter the following command:

  SITE RECFM(FB) LRECL(80) BLKSIZE(8000)

Step 3: Re-convert(import) the SAS transport file to a SAS data set on MVS

After transferring the SAS transport file, you may use the following sample on the mainframe to re-convert the SAS transport file to a SAS data set.

```
//iii JOB (aaaa,,class),name
//   JCLLIB ORDER=ZABCRUN.PROCLIB
//  EXEC SAS
//TRANS  DD DSN=aaaaiii.dsname1,DISP=SHR,
//       DCB=(RECFM=FB,LRECL=80,BLKSIZE=8000)
//OUT1 DD DSN=aaaaiii.dsname2,DISP=(NEW,CATLG),
//       UNIT=FILE,SPACE=(TRK,(s1,s2))
//SYSIN DD *

LIBNAME TRANS  XPORT;
PROC COPY IN = TRANS  OUT = OUT1 ;
```

4. Upload SAS Library with member-type CATALOG from PC-environment to MVS

Please Note:
If the SAS Library contains member-type CATALOG then you must use PROC CPORT to create a SAS transport file.

Step 1: Create a SAS transport file on the PC environment.

```
LIBNAME   STD1   ’C:\mydir’;
FILENAME  TRANS  ’C:\mydir\filename’  ;
PROC CPORT LIBRARY = STD1  FILE = TRANS ;
RUN;
```

Step 2: Transfer the SAS transport file

a) if you transfer the file by way of a software communication package(like KERMIT, ProComm or other) to the MVS environment be certain to specify the following parameters...

```
SET DATA BINARY
SET RECFM FB
SET LRECL 80
SET BLKSIZE 8000
```

b) if you transfer the SAS transport file via FTP and you are using WIN95 or WIN NT then
- Select **Binary** under the MODE menu(located in the center of the FTP WINDOW)
- Select **"send quoted command to remote"** from the COMMAND menu in the FTP window and enter the following command:

SITE RECFM(FB) LRECL(80) BLKSIZE(8000)

Step 3: Re-convert(import) the SAS transport file to a SAS data set on MVS

After transferring the SAS transport file, you may use the following sample on the mainframe to re-convert the SAS transport file to a SAS data set.

```
//iii JOB (aaaa,,E),name
//   JCLLIB ORDER=ZABCRUN.PROCLIB
//  EXEC SAS
//TRANS DD DSN=aaaaiii.transport.file,DISP=SHR,
//         DCB=(RECFM=FB,LRECL=80,BLKSIZE=8000)
//OUT1  DD DSN=aaaaiii.v6sas.formats,DISP=(NEW,CATLG),
//      UNIT=FILE,SPACE=(TRK,(s1,s2))
//SYSIN DD *
```

Or use LIBNAME and FILENAME statements rather than the JCL (DD statements)

```
FILENAME TRANS ’aaaaiii.transport.file’ RECFM=FB LRECL=80
                BLKSIZE=8000 DISP=SHR ;
LIBNAME OUT1 ’aaaaiii.v6sas.formats’
              UNIT=FILE DISP=(NEW,CATLG) SPACE=(TRK,(s1,s2)) ;

PROC CIMPORT LIBRARY = OUT1 INFILE = TRANS ;
```

How to… Transform SAS data set on MVS to an SPSS portable file

1. Convert SAS version 6 data set to version 5
2. Convert SAS version 5 data set to a standard SPSS file
3. Convert a standard SPSS file to an SPSS portable file

1. Convert SAS version 6 data set on disk to version 5

```
//iii  JOB  (aaaa,bbb,class),name
//   JCLLIB ORDER=ZABCRUN.PROCLIB
//  EXEC  SAS
//IN1  DD  DSN=aaaaiii.v6dsname,DISP=SHR
//OUT1  DD DISP=(NEW,CATLG),SPACE=(CYL,(s1,s2),RLSE),UNIT=FILE,
//   DSN=aaaaiii.v5dsname,DCB=(DSORG=DA,RECFM=U)
//SYSIN  DD  *

/* specify the V5 engine in the LIBNAME statement */

LIBNAME OUT1 V5  ;

PROC COPY IN = IN1 OUT = OUT1 ;

PROC CONTENTS DATA = OUT1._ALL_ ;
```

2. Convert SAS version 5 data set to a standard SPSS file

```
//iii  JOB  (aaaa,bbb,class),name
//   JCLLIB ORDER=ZABCRUN.PROCLIB
//   EXEC SPSS
//IN1  DD DISP=SHR,DSN=aaaaiii.v5dsname
//OUT1  DD UNIT=FILE,DISP=(NEW,CATLG),
//   DSN=aaaaiii.spssdsname,
//   SPACE=(TRK,(s1,s2),RLSE)
//SYSIN DD *

   /* this method applies to version 5 SAS data set */

GET SAS DATA = IN1.SASdatasetname

SAVE OUTFILE = OUT1
```

3. Convert a standard SPSS file to an SPSS portable file

```
//iii  JOB  (aaaa,bbb,class),name
//   JCLLIB  ORDER=ZABCRUN.PROCLIB
//   EXEC  SPSS
//IN1  DD  DSN=aaaaiii.spssdsname,DISP=SHR
//OUT1  DD DSN=aaaaiii.transp,DISP=(NEW,CATLG),UNIT=FILE,
//   DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200),SPACE=(TRK,(s1,s2))
//SYSIN  DD  *
GET FILE = IN1
EXPORT  OUTFILE = OUT1
```

How to… Download an SPSS portable file to the PC environment

1. **Download an SPSS portable file to the PC**
2. **Re-convert(import) the SPSS portable file to a standard SPSS file on the PC**

1.Download an SPSS portable file to the PC

Please note:
An SPSS portable file is an ASCII character file and not a binary file

a) if you transfer the file by way of a software communication package(like KERMIT, ProComm… or other) to the PC environment, you must specify the following parameters:

```
SET DATA ASCII
SET RECFM FB
SET LRECL 80
SET BLKSIZE 3200
```

b) if you transfer the SPSS portable file via FTP and you are using WIN95 or WIN NT then
- Select **ASCII** under the MODE menu(located in the center of the FTP WINDOW)
- Select **"send quoted command to remote"** from the COMMAND menu in the FTP WINDOW and enter the following command:

```
SITE RECFM(FB) LRECL(80) BLKSIZE(3200)
```

3. Re-convert(import) an SPSS portable file to a standard SPSS file on the PC

After transferring the SPSS portable file, use the following SPSS statement on the PC to re-convert the SPSS portable file to a standard SPSS file.

In SPSS for Windows, follow these steps:

1. from the **File** menu choose **Open** then **Data**
2. in the **Open Data File** window choose **SPSS portable (*.por**) from the **File Type:** field
3. locate the **SPSS** portable file
4. click the **OK** button

To save the file as a standard SPSS file follow these steps:

1. from the **File** menu choose **Save As**
2. from the **Save file as** type field choose **SPSS (*.sav)**
3. enter a file name in the **File name:** box
4. click the **OK** button

How to… Transform standard SPSS file on PC to a SAS data set on MVS

1. Create an SPSS portable file on the PC
2. Transfer an SPSS portable file to the mainframe
3. Re-convert(import) an SPSS portable file to a standard SPSS file on MVS
4. Convert the standard SPSS file to a SAS data set on MVS

1. Create an SPSS portable file on the PC

Windows Environments:

First open your **SPSS** save file using **SPSS** for Windows.

To create an **SPSS** portable file follow these steps:

1.from the **File** menu choose **Save As**
2.from the **Save file as type** field choose **SPSS portable (*.por)**
3.enter a file name in the **File name**: box
4.click the **OK** button
An SPSS portable file is a **text** file. If you are transferring this file electronically do not transfer it as a binary file.

2. Transfer an SPSS portable file to the mainframe

Please note:
An SPSS portable file is an ASCII character file and not a binary file

a) if you transfer the file by way of a software communication package(like KERMIT, ProComm… or other) to the PC environment be certain to specify the following parameters:

```
SET DATA ASCII
SET RECFM FB
SET LRECL 80
SET BLKSIZE 3200
```

b) if you transfer the SPSS portable file via FTP and you are using WIN95 or WIN NT then

- Select **ASCII** under the MODE menu(located in the center of the FTP window)
- Select **"send quoted command to remote"** from the COMMAND menu in the FTP window and enter the following command:

  SITE RECFM(FB) LRECL(80) BLKSIZE(3200)

3. Re-convert(import) an SPSS portable file to an standard SPSS file on MVS

After transferring the SPSS portable file, you may use the following sample on the mainframe to re-convert the SPSS portable file to a standard SPSS file.

```
//iii JOB (aaaa,,class),name
//   JCLLIB ORDER=ZABCRUN.PROCLIB
//  EXEC SPSS
//TRANS  DD DSN=aaaaiii.spss.portable,DISP=SHR,
//       DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
//OUT1 DD DSN=aaaaiii.spss.system,DISP=(NEW,CATLG),
//       UNIT=FILE,SPACE=(TRK,(s1,s2))
//SYSIN DD *
IMPORT FILE = TRANS
SAVE OUTFILE = OUT1
```

4.Convert the standard SPSS file to a SAS data set on MVS

```
//iii JOB (aaaa,,class),name
//   JCLLIB ORDER=ZABCRUN.PROCLIB
//  EXEC SAS
//IN1  DD DSN=aaaaiii.spss.system,DISP=SHR
//SYSIN DD *

      /* does not apply to SAS version 5 */

LIBNAME   IN1  SPSS ;
LIBNAME  OUT1  ’aaaaiii.dsname UNIT=FILE DISP=(NEW,CATLG) SPACE=(TRK,(s1,s2)) ;

PROC CONVERT  SPSS = IN1  OUT = OUT1 ;
```

Or process a standard SPSS file with a SAS DATA Step

```
//iii  JOB  (aaaa,bbb,class),name
//   JCLLIB  ORDER=ZABCRUN.PROCLIB
//   EXEC SAS
//IN1   DD  DSN=aaaaiii.spssdsname,DISP=SHR
//SYSIN  DD  *

  /* Since SPSS files do not have an internal member name
     as in SAS; an SPSS file is referred to as _FIRST_ */

Libname IN1 SPSS ;
DATA TEST ;
  SET IN1._FIRST_ ;
```

Or process a standard SPSS file with SAS procedures

```
//iii JOB (aaaa,,class),name
//   JCLLIB ORDER=ZABCRUN.PROCLIB
//  EXEC SAS
//IN1  DD DSN=aaaaiii.spss.system,DISP=SHR
//SYSIN DD *

LIBNAME   IN1  SPSS ;

PROC MEANS  DATA = IN1._FIRST_;
     VAR  variable list   ;

PROC CONTENTS DATA=IN1._FIRST_  ;
```

How to… Copy SAS Data Library from a storage device to another

The user is responsible for backing up ones own data sets using SAS procedures. There are two procedures used to copy SAS data set: PROC COPY and PROC DATASETS. Refer to the SAS Procedures Guide for details.

Note: A word of caution!
You can use the standard IBM DFSMS/MVS utility IEBGENER to copy or move SAS data sets that were created with the V6 engine or with the V6SEQ engine. As long as the block size of the data library that you are copying is not greater than the track capacity of the target device, a V6 or V6SEQ data library may be transferred between unlike device types(for example, 3350 to 3390).

However, we caution the user. In a few cases it has been observed that the IBM/MVS utility did not produce usable copies of SAS data sets. SAS Institute ***encourages*** *the user to use the SAS procedures to copy or move SAS data libraries from one file to another or from one storage device to another.*

Example 1:

Copies all members in the SAS data library referenced by libref IN1 on FILE to a library referenced by the *libref* OUT1 on a NIH standard tape.

```
//  (JOB statement)
//  JCLLIB ORDER=ZABCRUN.PROCLIB
//  EXEC SAS
//SYSIN DD *

LIBNAME IN1   'aaaaiii.dsname1';
LIBNAME OUT1  'aaaaiii.dsname2'   DISP=(NEW,CATLG)  UNIT=TAPE
   VOL=SER=nnnnnn  LABEL=(seqnum,lab) ;

PROC COPY IN = IN1  OUT = OUT1;
```

Or use the JCL DD statements

```
//  (JOB statement with class B or C)
/*ROUTE XEQ TAPE
/*MESSAGE nnnnnn,W
//  JCLLIB ORDER=ZABCRUN.PROCLIB
//  EXEC SAS
//IN1  DD DSN=aaaaiii.dsname1,DISP=SHR
//OUT1 DD UNIT=TAPE,DSN=aaaaiii.dsname2,
//  DISP=(NEW,CATLG),VOL=SER=nnnnnn,LABEL=(seqnum,lab)
//SYSIN DD *
```

Example 2:

Copy selected member REGION1 in the SAS data library referenced by libref IN1 on FILE to a library referenced by the *libref* OUT1 on a NIH standard tape.

```
//  (JOB statement)
//  JCLLIB ORDER=ZABCRUN.PROCLIB
//  EXEC SAS
//SYSIN DD *

LIBNAME IN1   'aaaaiii.dsname1';
LIBNAME OUT1  'aaaaiii.dsname2'   DISP=(NEW,CATLG)  UNIT=TAPE
  VOL=SER=nnnnnn  LABEL=(seqnum,lab) ;

PROC COPY IN = IN1  OUT = OUT1;
   SELECT REGION1 ;
         /* to copy selected members use a SELECT statement */
```

How to…convert a version 5 SAS library to version 6

Method 1:

If the SAS library contains many members (e.g. many SAS data sets) then specify the SELECT statement and name a member.

```
//iii  JOB (aaaa,bbb,A),name
//  JCLLIB ORDER=(ZABCRUN.PROCLIB)
// EXEC SAS
//IN1  DD DSN=aaaaiii.dsname1,DISP=SHR
//OUT1 DD DSN=aaaaiii.dsname2,
//   DISP=(NEW,CATLG),UNIT=FILE,SPACE=(TRK,(s1,s2),RLSE)
//SYSIN DD *

PROC V5TOV6 IN = IN1 OUT = OUT1 ;
  SELECT SASdatasetname;

PROC CONTENTS DATA = OUT1.SASdatasetname ;
```

Method 2:

You can also upgrade a SAS data set to version 6 by using PROC COPY. The program below reads the version 5 SAS data set MEDHIST from the SAS library aaaaiii.STUDY105.SASLIB and creates a new SAS library that is a version 6 library.

By default, when SAS version 6 runs, it assumes that the new SAS libraries should be stored as version 6 unless stated otherwise. For this reason you don’t need to specify the version in the second LIBNAME statement.

```
  //iii JOB (aaaa,box,class),lastname
  //  JCLLIB ORDER=ZABCRUN.PROCLIB
  //  EXEC SAS
  //SYSIN DD *

  LIBNAME IN1 ’aaaaiii.STUDY105.SASLIB’ disp=shr;
  LIBNAME OUT1 ’aaaaiii.STUDY105.SASLIB.NEW’ disp=(new,catlg)
              unit=file space=(trk,(1,10));

  PROC COPY IN = IN1 OUT = OUT1;
    SELECT  MEDHIST;
```

How to…convert a version 6 SAS library to version 5

```
//iii  JOB  (aaaa,bbb,class),name
//   JCLLIB ORDER=ZABCRUN.PROCLIB
//  EXEC  SAS
//IN1  DD  DSN=aaaaiii.v6dsname,DISP=SHR
//OUT1  DD DISP=(NEW,CATLG),SPACE=(CYL,(s1,s2),RLSE),UNIT=FILE,
//   DSN=aaaaiii.v5dsname,DCB=(DSORG=DA,RECFM=U)
//SYSIN  DD  *

     /* specify the V5 engine in the LIBNAME statement */
        If the SAS library contains many members then specify
        the SELECT statement and name a member */

     LIBNAME OUT1 V5  ;

     PROC COPY IN = IN1 OUT = OUT1 ;

     PROC CONTENTS DATA = OUT1._ALL_ ;
```

PROC SORT - Sorting Permanent SAS Data Sets

When sorting a permanent SAS data set, always use the OUT = option on the PROC SORT statement. Without this option, SAS will attempt to sort the data set on permanent disk space.

```
//  (JOB statement)
//  JCLLIB ORDER=ZABCRUN.PROCLIB
//  EXEC SAS
//ddname DD DSN=aaaaiii.dsname,DISP=SHR
//SYSIN DD *

PROC SORT DATA = ddname.sasdsname OUT=newsasdsname ;
```

PROC CONTENTS and PROC DATASETS Procedures

There are two SAS procedures that examine the descriptor portion of a SAS data set. Whenever a user creates a permanent SAS data set, documentation should also be prepared. The documentation should provide sufficient means for someone to determine what information the file contains and how to get it. Both the **PROC CONTENTS** and **PROC DATASETS** procedures have some nice ways to supplement hand-prepared documentation.

The CONTENTS procedure lists the names of SAS data sets in a library and information from the directory of a SAS data set. The main CONTENTS results are presented in a table, showing in alphabetical order by variable name each variable’s type and length, position in the SAS data set vector, permanent format and informat if any, and label if any.

PROC CONTENTS Procedure

Examples:

(1) Print the contents of a temporary SAS data set called TEST1

```
PROC CONTENTS DATA = TEST1;
```

(2) Print the contents of a permanent SAS data set called MASTER

```
LIBNAME libref  ’aaaaiii.dsname’  ;

PROC CONTENTS DATA = libref.MASTER;
```

(3) If you do not remember the second-level name of a SAS data set, use the _ALL_ SAS keyword.

```
LIBNAME libref  ’aaaaiii.dsname’  ;

PROC CONTENTS DATA = libref._ALL_;
```

PROC DATASETS Procedure

With Version 6 of the SAS System, the DATASETS procedure was made to subsume the functions of several utility procedures including the CONTENTS procedure. Note that the CONTENTS statement of the DATASETS procedure provides the same function as the CONTENTS procedure.

Examples:

(1) Obtain a directory of all members in a SAS library

```
               LIBNAME  libref  'aaaaiii.dsname' ;
               PROC DATASETS  LIBRARY = libref   ;
```

LIBRARY = *libref* processes the permanent SAS library. If omitted, the default library WORK is processed.

If you do not remember the second-level name of a SAS data set, use the _ALL_ SAS keyword.

```
               LIBNAME  libref   'aaaaiii.dsname' ;
               PROC DATASETS LIBRARY =  libref ;
                  CONTENTS  DATA = libref._ALL_ ;
```

The PROC DATASETS procedure

- lists, renames, and deletes SAS files see the SAS Procedures Guide
- alters variable names in the SAS data set (cannot alter a variable's type or length) see the SAS Procedures Guide
- cannot be used with SAS libraries on tape
- provides all the capabilities of the APPEND, CONTENTS, and COPY procedures
- enhancements include options that assign, change, and remove passwords for members of SAS data libraries.

Note:
The DATASETS procedure provides all the capabilities of the APPEND, CONTENTS, and COPY procedures, as well as the customary PROC DATASETS capabilities. The APPEND, CONTENTS, and COPY procedures continue to be supported, so existing SAS programs that use these procedures do not require modifications.

Various Statements Used with the PROC DATASETS Procedure

The PROC DATASETS procedure provides many functions by using the CONTENTS, CHANGE, and DELETE statements.
Please see a detailed summary of the PROC DATASETS functions in the SAS manual: SAS Procedures Guide

CONTENTS statement

Purpose: To list the contents of a SAS data set

Example 1:

```
LIBNAME IN1 ’WXYZABC.PERM.SASDATA’  ;
PROC DATASETS  LIBRARY = IN1  ;
  CONTENTS  DATA = SMOKERS ;
```

Example 2:

```
LIBNAME IN1 ’WXYZABC.PERM.SASDATA’ ;
PROC DATASETS  LIBRARY = IN1 ;
  CONTENTS  DATA = IN1._ALL_ ;
```

When you specify the name of a member, you get a report on that member. When you specify _ALL_ , you get information about all members.

CHANGE Statement

Purpose: To rename a SAS data set

```
General Form:     LIBNAME libref  ’aaaaiii.dsname’ ;
                  PROC DATASETS   LIBRARY = libref ;
                     CHANGE  oldname1 = newname1 ;
```

Example:

NOTE: Specify option DISP=OLD in the LIBNAME statement to rename or delete SAS files.

```
                  LIBNAME IN1 ’WXYZABC.PERM.SASDATA’  DISP=OLD ;
                  PROC DATASETS  LIBRARY  =  IN1  ;
                    CHANGE  SMOKERS  =  NOSMOKER ;
```

DELETE statement

Purpose: To delete selected member of a SAS data library.

```
General Form:     LIBNAME libref ’aaaaiiii.dsname’  DISP=OLD ;
                  PROC DATASETS   LIBRARY = libref ;
                    DELETE   member list/options ;
```

Example:

NOTE: Specify option DISP=OLD in the LIBNAME statement to rename or delete SAS files.

```
                  LIBNAME IN1 ’WXYZABC.PERM.SASDATA’ DISP=OLD;
                  PROC DATASETS  LIBRARY = IN1  ;
                    DELETE  PLACEBO ;
```

SAS Format Library

User-written formats and informats are stored in a type of SAS file called a *catalog.*

User-written informats and formats are either temporary or permanent, depending on whether they are stored in a temporary or permanent catalog.

Temporary Formats and Informats:

Formats and informats are temporary when you **do not** specify the **LIBRARY =** option in the PROC FORMAT statement. By default, the user-defined formats are stored in a temporary catalog named WORK.FORMATS. Consequently these formats disappear when SAS terminates. Therefore, you can use temporary informats and formats only in the same SAS job or session in which they are created.

Example:

```
PROC FORMAT ;  /* No LIBRARY = option, so formats and informats
                  stored in the catalog WORK.FORMATS  */
```

Permanent Formats and Informats:

Permanent format catalogs can be stored on disk, the MSS, or tape, but they must be used from disk or the MSS.

Example 1:

```
LIBNAME  LIBRARY  'aaaaiii.dsname' DISP=(NEW,CATLG) SPACE=(TRK,(s1,s2)) UNIT=FILE
;

PROC FORMAT  LIBRARY=LIBRARY ; /* Formats or informats stored in a catalog
                                  named FORMATS in the library LIBRARY.
                                  It is recommended, but not required, that
                                  you use the word LIBRARY as the libref */
```

Example 2:

```
LIBNAME  abc  'aaaaiii.dsname' DISP=(NEW,CATLG) SPACE=(TRK,(s1,s2)) UNIT=FILE;


PROC FORMAT  LIBRARY= abc ;    /* Formats or informats stored in a catalog
                                  named FORMATS in the library abc. If there
                                  isn't a catalog named FORMATS in the library
                                  abc, the SAS System creates one.*/
```

Example 3:

```
LIBNAME  out1 'aaaaiii.dsname' DISP=(NEW,CATLG) SPACE=(TRK,(s1,s2)) UNIT=FILE;

PROC FORMAT  LIBRARY= way.cool ; /* Formats or informats stored in a
                                    catalog named cool in the library
                                    way */
```

A Sampling of Programs to Create, Use and Convert SAS Format Libraries

Create a permanent format catalog on disk

```
//  (JOB statement)
//  JCLLIB ORDER=ZABCRUN.PROCLIB
//  EXEC SAS
//LIBRARY DD UNIT=FILE,DISP=(NEW,CATLG),
//  DSN=aaaaiii.dsname,SPACE=(TRK,(s1,s2))
//SYSIN DD *

PROC FORMAT LIBRARY=LIBRARY ;
     value   format name
       (enter specifications)
     ;
```

- **Or with the LIBNAME statement**

```
//  (JOB statement)
//  JCLLIB ORDER=ZABCRUN.PROCLIB
//  EXEC DSSCR,NAME='aaaaiii.dsname'
//  EXEC SAS
//SYSIN DD *

LIBNAME LIBRARY 'aaaaiii.dsname' UNIT=FILE  DISP=(NEW,CATLG)
SPACE=(TRK,(s1,s2)) ;

PROC FORMAT LIBRARY=LIBRARY ;
     value   format name
       (enter specifications)
     ;
```

Use formats from a permanent format catalog on disk

Permanent formats and informats can be used in the program that creates them and in subsequent SAS jobs or sessions.
In DATA and PROC steps following the PROC FORMAT step, the SAS System searches the **LIBRARY** library for the **FORMATS** catalog file.

```
//  (JOB statement)
//  JCLLIB ORDER=ZABCRUN.PROCLIB
//  EXEC SAS
//LIBRARY DD DSN=aaaaiii.dsname,DISP=SHR
//SYSIN DD *
PROC FORMAT LIBRARY=LIBRARY ;
```

- **Or with the LIBNAME statement**

```
//  (JOB statement)
//  JCLLIB ORDER=ZABCRUN.PROCLIB
//  EXEC SAS
//SYSIN DD *

LIBNAME LIBRARY 'aaaaiii.dsname' ;
PROC FORMAT LIBRARY=LIBRARY FMTLIB ;
```

If you wish to review the entries in the format library the **FMTLIB** option may be used to display the contents of a permanent format catalog

A Sampling of Programs to Create, Use, and Convert Version 5 SAS Format Libraries

Reminders to those who still have permanent Version 5 format libraries.

- Permanent Version 5 format libraries are partitioned data sets.
- To copy version 5 format libraries, use PROC PDSCOPY.
- While Version 5 format libraries can be stored on tape, they must be on disk to be used in a SAS program.
- Version 5 formats can be used without conversion in Version 6 programs.
- Version 6 formats cannot be used in Version 5 programs.

Create a permanent Version 5 SAS format library on disk using a Version 5 program

```
//  (JOB statement)
//  JCLLIB ORDER=ZABCRUN.PROCLIB
//  EXEC SAS518
//SASLIB DD UNIT=FILE,DISP=(NEW,CATLG),
//  DSN=aaaaiii.dsname,SPACE=(TRK,(s1,,d1))
//SYSIN DD *

 PROC FORMAT LIBRARY = SASLIB ;
```

where ’d1,’ the directory parameter, is set equal to the number of formats output divided by 4.

Use permanent Version 5 SAS format library on disk

```
//  (JOB statement)
//  JCLLIB ORDER=ZABCRUN.PROCLIB
//  EXEC SAS
//SASLIB DD DSN=aaaaiii.dsname,DISP=SHR
//SYSIN DD *

 PROC FORMAT LIBRARY = SASLIB ;
```

Version 5 SAS data sets and SAS format libraries can be used without conversion in a Version 6 program. However, you may choose to convert the Version 5 SAS format library to Version 6 SAS format library using the V5TOV6 procedure.

PROC V5TOV6 procedure converts Version 5 SAS format library to Version 6 SAS format library

```
//iii  JOB (aaaa,,E),name
//   JCLLIB ORDER=ZABCRUN.PROCLIB
//  EXEC  SAS
//SASLIB DD DSN=aaaaiii.v5sas.formats,DISP=SHR
//LIBRARY DD UNIT=FILE,DISP=(NEW,CATLG),
//  DSN=aaaaiii.v6sas.formats,SPACE=(TRK,(s1,s2))
//SYSIN DD *

PROC V5TOV6 FORMAT=SASLIB OUT=LIBRARY;
```

How to… Transform a Version 5 SAS Format Library to a Version 6 and Download to the PC Environment.

1. Convert Version 5 SAS format library to Version 6 SAS format library
2. Create a SAS transport file
3. Transfer(download) the SAS transport file
4. Re-convert(import) the SAS transport file to a Version 6 SAS format library on the PC

1. Convert Version 5 SAS format library to Version 6 SAS format library

```
//iii  JOB (aaaa,,E),name
//   JCLLIB ORDER=ZABCRUN.PROCLIB
//  EXEC  SAS
//SASLIB DD DSN=aaaaiii.v5sas.formats,DISP=SHR
//LIBRARY DD UNIT=FILE,DISP=(NEW,CATLG),
//  DSN=aaaaiii.v6sas.formats,SPACE=(TRK,(s1,s2))
//SYSIN DD *
PROC V5TOV6 FORMAT=SASLIB OUT=LIBRARY;
```

2. Create SAS transport file

```
//iii JOB (aaaa,,E),name
//   JCLLIB ORDER=ZABCRUN.PROCLIB
//  EXEC SAS
//STD1 DD DSN=aaaaiii.v6sas.formats,DISP=SHR
//TRANFILE DD DSN=aaaaiii.transport.file,DISP=(NEW,CATLG),
//  UNIT=FILE,DCB=(RECFM=FB,LRECL=80,BLKSIZE=8000),
//  SPACE=(TRK,(s1,s2))
//SYSIN DD *
PROC CPORT LIBRARY=STD1 FILE=TRANFILE ;
```

• **Or** with the LIBNAME statement

```
LIBNAME STD1 ’aaaaiii.v6sas.formats’;
FILENAME TRANFILE ’aaaaiii.transport.file’
  UNIT=FILE DISP=(NEW,CATLG)  SPACE=(TRK,(s1,s2))
  RECFM=FB LRECL=80 BLKSIZE=8000 ;

PROC CPORT LIBRARY=STD1 FILE=TRANFILE ;
```

3. Transfer(download) the SAS transport file

a) if you transfer the file by way of a software communication package(like KERMIT, ProComm or other) to the PC environment be certain to specify the following parameters...

```
SET DATA BINARY
SET RECFM FB
SET LRECL 80
SET BLKSIZE 8000
```

c) if you transfer the SAS transport file via FTP and you are using WIN95 or WIN NT then
- Select Binary under the MODE menu(that’s located in the center of the FTP WINDOW)
- Select "send quoted command to remote" from the COMMAND menu in the FTP window and enter the following command:

```
SITE RECFM(FB) LRECL(80) BLKSIZE(8000)
```

4. Re-convert(import) the SAS transport file to a SAS format library on the PC
After transferring the SAS transport file, you may use the following sample on the PC to re-convert(import) the SAS transport file to a SAS format library.

```
FILENAME TRANFILE  ’C:\mydir\filename’;
LIBNAME  OUT1  ’C:\mydir’ ;
PROC CIMPORT LIBRARY = OUT1 INFILE = TRANFILE ;
RUN;
```

The PROC CATALOG Procedure and SAS Format Libraries

SAS member-type catalog is used by the SAS software products Base SAS, SAS/FSP, SAS/AF, SAS/IML, and SAS/GRAPH. In Base SAS the PROC CATALOG procedure is used to manage entries in SAS member-type catalogs.

Answers to commonly asked questions relating to SAS Format Libraries(member-type catalog).

How to… add to or modify a SAS Format Library

And in most cases you do not need to re-run the original program that generated the SAS format library.

There is no problem with adding or modifying a format library, however, there is a problem modifying an entry(i.e. a format) within the format library simply because a format is a compiled entry.

Write a new SAS program using the PROC FORMAT procedure and reference the original SAS format library in the LIBNAME statement.

```
LIBNAME  OLD  ’aaaaiii.DSNAME’  DISP = (OLD,CATLG)  UNIT=FILE ;

PROC FORMAT LIBRARY= OLD ;
   VALUE    /* enter the format name
               and the formats */
   ;
```

a) If you create a format with the same format name currently in the library, then it will simply over-write it.

b) If the format name is not the same, then it will be added to the permanent SAS format library.

c) In some cases you may encounter a limitation…one cannot increase the SPACE allocation to a permanent SAS Library on MVS.

How to… combine SAS Format Libraries

You may want to create a new SAS format library and then combine the two libraries, that is the original with the new one.

Example:

```
Libname library ’aaaaiii.FMTSET2’;
Libname combine ’aaaaiii.FMTSET1’;
proc catalog cat=library.FORMATS;
copy out=combine.FORMATS;
```

How to…use formats created in a permanent SAS format library

Method 1:

Use formats created in a permanent SAS format library by including a LIBNAME statement with a libref LIBRARY

```
//  (JOB statement)
//  JCLLIB ORDER=ZABCRUN.PROCLIB
//  EXEC SAS
//SYSIN DD *

libname library 'aaaiii.dsname';
  proc freq data=library.demog;
    table race*sex;
    format race race. sex sex.;
```

Method 2:

In general, you can use any libref in the LIBNAME statement.
If you do not use the libref LIBRARY then use the SAS option FMTSEARCH=
to specify the libref you used.

```
//  (JOB statement)
//  JCLLIB ORDER=ZABCRUN.PROCLIB
//  EXEC SAS
//SYSIN DD *

libname  in  'aaaaiii.dsname';
  options  fmtsearch = (in);
  proc freq data=in.demog;
     table race*sex;
     format race race. sex sex.;
```

Method 3:

To access more than one format library include one LIBNAME statement
for each library then list the librefs after the FMTSEARCH= option separated by spaces.

```
  //  (JOB statement)
  //  JCLLIB ORDER=ZABCRUN.PROCLIB
  //  EXEC SAS
  //SYSIN DD *

   libname in1 'aaaaiii.dsname1';
   libname in2 'aaaaiii.dsname2';
  options  fmtsearch = (in1 in2);
```

PROC PRINTTO Procedure

PROC PRINTTO is used to route the output of SAS procedures to an external file rather than to the SAS procedure output file. The following program will put the results of the PROC PRINT into an external file called ’dsname1’ and the results of the PROC MEANS in the usual SAS procedure output file.

```
//  (JOB statement)
//  JCLLIB ORDER=ZABCRUN.PROCLIB
//  EXEC DSSCR,NAME=’aaaaiii.dsname1’
//  EXEC SAS
//ddname1 DD UNIT=FILE,DISP=(NEW,CATLG),
//  DSN=aaaaiii.dsname1,SPACE=(TRK,(s1,s2)),
//  DCB=(RECFM=FB,LRECL=len,BLKSIZE=b)
//ddname2 DD DSN=aaaaiii.dsname2,DISP=SHR
//SYSIN DD *

PROC PRINTTO NEW FILE=ddname1 ;

PROC PRINT DATA = ddname2.sasdsname ;

PROC PRINTTO ;

PROC MEANS DATA = ddname2.sasdsname ;
```

NOTE:
PROC PRINTTO procedure will not be applicable in Version 7.

In Version 7 the Output Delivery System(ODS), a new SAS subsystem, provides capabilities for displaying and controlling the output from the SAS procedures. Therefore, Version 7 release of SAS will not have the PROC PRINTTO procedure.

Efficient SAS Programming - A Few Tips

How to… save memory, CPU time, I/O operations, storage, and programmer time.

1. **Read only the fields you need.**
   When you read an external file, you spend CPU time to read a field. If you use list input without pointer controls, you must read every field in an external file. When you use list input with pointer controls, column input or formatted input, you exclude unneeded variables from your data set and spend less CPU time reading fields you actually do not need.

2. **Store data in SAS data sets.**
   When you need to use the SAS System repeatedly to analyze or manipulate any particular group of data, create a permanent SAS data set instead of reading the raw data each time to create a temporary SAS data set.

3. **Process only the variables you need.**
   When you refer to an SAS data set using the SET, MERGE, or UPDATE statement, select the variables you need using the DROP= or KEEP= data set option rather than the DROP or KEEP statement. You can use the DROP or KEEP statement in combination with the DROP= or KEEP= data set option to eliminate variables created in the current DATA step from the output data set.

4. **Take advantage of SAS procedures.**
   Use procedures to examine your data and know the characteristics of your data. For instance, use (a) PROC CHART with the HBAR statement to see the big picture, (b) PROC PLOT to spot relationships among data variables, (c) PROC MEANS to see variable ranges and limits, (d) Output SAS data sets from CORR, FREQ, and MEANS procedures as input to other procedures to avoid reading a master data set again.

5. **Use a WHERE statement or a WHERE= data set option.**
   Use a WHERE statement or a WHERE= data set option to run any procedure on a subset of a SAS data set. When you want to run a procedure with only the observations in a data set that meet specific conditions, use the WHERE statement to select those observations rather than creating a subset data set and then running the procedure.

6. **If appropriate, specify the reserved name _NULL_ in the DATA statement to have the SAS System go through a DATA step without creating a SAS data set.**
   For instance, you may need to write reports with PUT statements.

7. **Sorting is relatively expensive in terms of CPU time and may require large work space. Therefore, plan sorting to reduce the number of sorts.**

   a) If appropriate, use two or more variables in the BY statement of one PROC SORT step rather than using two PROC SORT steps, each with one BY variable.
   b) If a procedure supports both the CLASS statement and the BY statement, use the CLASS statement to eliminate the need for sorted input.
   c) Remove unneeded variables and observations either in a DATA step before the PROC SORT step, or use the DROP= or KEEP= data set option and the WHERE statement in the PROC SORT step.

6. **Programs that fail due to preventable errors waste resources.**

   Using the OBS= option with the INFILE statement to read a test raw file, and the OBS= data set option with the SET statement allows the user to use a subset of actual production data to test a program.

9. **If you do not plan to perform arithmetic operations on variables containing digits, store the variables as character rather than numeric.**

10. Use the LENGTH statement to reduce the storage space for variables in SAS data sets.

Use the LENGTH statement to assign lengths to your numeric variables. Numeric values (no matter how many digits they contain) are stored as floating point numbers in 8 bytes of storage unless you specify another length. For most applications, 4 bytes is sufficient. Changing from 8 to 4 bytes will half the storage requirements and speed up data transfer. When numeric variables are assigned lengths less than 8 bytes, all data step calculations are still done with double precision arithmetic.
You can also assign lengths to character variables; however, if they were read in with formats or columns specified, they will already have their lengths defined.

Purpose: To define the number of bytes used to store values of variables in the SAS data set.

```
General Form:     LENGTH variable(s)  [$]  length...  [DEFAULT=n] ;
                                       |        |              |
                                      (A)      (B)            (C)
```

(A) $	indicates the preceding variable or variables are character variable(s)
(B) Length in Bytes	In the MVS environment this constant represents a range from 2 to 8 for numeric variables and 1 to 200 for character variables In the Windows, Mac, and UNIX environments this constant represents a range from 3 to 8 for numeric variables and 1 to 200 for character variables
(C) n	changes the default number of bytes used for storing the values of newly created numeric variables from 8 (the default length) to the value of n(n can range from 2 to 8 or 3 to 8, depending on the host system)

The following table shows the largest integer represented for the corresponding Length in Bytes. Limits shown in this table are for the MVS operating system. Limits are different for Windows, Mac, and Unix. (Please refer to the appropriate SAS Companion guide).

Length in Bytes	**Largest Integer Represented**
2	255
3	65,535
4	16,777,215
5	4,294,967,295
6	1,099,511,627,775
7	281,474,946,710,655
8	72,057,594,037,927,935

For example, if you know that a numeric variable always has values between 0 and 100, you can use a length of 2 to store the number and save space on storage.
Note:
You can safely use the LENGTH statement when the values for the variable are integers. For non-integers (fractional values) it is highly recommended that you do not use the LENGTH statement.
Advantages:

- Save space on storage
- Saves time in reading and writing the data
- Changes SAS defaults for storing the values

How to Invoke the SAS Macro Facility

To invoke the Macro Facility in SAS use the OPTINS='MACRO' option in the
// EXEC SAS statement

```
//  EXEC SAS,OPTIONS='MACRO'
```

For more information about the SAS Macro Facility, refer to the SAS Guide to Macro Processing.

How to Use Curly Braces in a SAS Program at NIH

The use of curly braces with explicitly sub-scripted arrays is not supported at NIH. You must use parentheses.

For instance, PROC IML procedure requires curly braces. To use curly braces one must type the following WYLBUR commands before entering the program:

```
            SET MAP #’8BC0’
            SET MAP #’9BD0’
```

If you use the PROC IML procedure often, enter these two **SET** commands in your **WYLBUR PROFILE**.

```
      ? USE PROFILE CLEAR (CR)
      ? LIST (CR)
      ? COLLECT
        1.? SET MAP #’8BC0’
        2.? SET MAP #’9BD0’
        3.? BREAK **
      ? SET PROFILE
      ? PROFILE DATA SET SAVED AND CATALOGUED
```

If you find the following ERROR message in the SAS Log of a PROC IML Step, **"ERROR 200-322: The symbol is not recognized"**, correct the problem with these instructions.

1) Return to the SAS program
2) Use the WYLBUR modify command:
   ? MOD   (and re-enter the curly braces)

2) Re-save the SAS program
3) **SET the WYLBUR PROFILE**… use the instructions shown(above)
5) 5) Re-run the SAS program